Improving Group Role Assignment Problem By Incremental Assignment Algorithm

by

Pinzhi Wang

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Improving Group Role Assignment Problem By Incremental Assignment Algorithm	
Name of Candidate Nom du candidat	Wang, Pinzhi	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance October 29, 2019

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Youssou Gningue
(Supervisor/Co-directeur de thèse)

Dr. Haibin Zhu
(Co-Supervisor/Co-directeur(trice) de thèse)

Dr. Matthias Takouda
(Committee member/Membre du comité)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
Monsieur David Lesbarrères
Dean, Faculty of Graduate Studies
Doyen, Faculté des études supérieures

Dr. Dongning Liu
(External Examiner/Examinateur externe)

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Pinzhi Wang**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Abstract

The Assignment Problem is a basic combinatorial optimization problem. In a weighted bipartite graph, the Assignment Problem is to find a largest sum of weights matching. The Hungarian method is a well-known algorithm which is combinatorial optimization.

Adding a new row and a new column to a weighted bipartite graph is called the Incremental Assignment Problem (IAP). The maximum weighted matching (the optimal solution) of the weighted bipartite graph has been given. The algorithm of the Incremental Assignment Problem utilizes the given optimal solution (the maximum weighted matching) and the dual variables to solve the matrix after extended bipartite graph.

This thesis proposes an improvement of the Incremental Assignment Algorithm (IAA), named the Improved Incremental Assignment Algorithm. The improved algorithm will save the operation time and operation space to find the optimal solution (the maximum weighted matching) of the bipartite graph.

We also present the definition of the Incremental Group Role Assignment Problem that based on the Group Role Assignment Problem (GRAP) and Incremental Assignment Problem (IAP). A solution has been designed to solve it by using the Improved Incremental Assignment Algorithm (IIAA).

In this thesis, simulation results are presented. We utilize the tests to compare the algorithm of the Incremental Assignment Problem and the Improved Incremental Assignment Algorithm (IIAA) to show the advantages of IIAA.

Keywords

Assignment Problem, Weighted Bipartite Graph, Hungarian Algorithm, Incremental Assignment Problem, Improved Incremental Assignment Algorithm, Group Role Assignment.

Acknowledgments

I would like to express my appreciation to my supervisors Dr. Youssou Gningue and Haibin Zhu. When I get confused, they always give me directions to find the way to solve the problem. They provide support and considerable time into this research, and many excellent suggestions to me. My thesis would be difficult to finish without their guidance and assistance.

I am so grateful to Dr. Matthias Takouda of accepting to be a member of the committee.

Also, I would like to thank my family members, especially my mother and my aunt. Without their support, I could not have a chance to be a graduate student in Laurentian University, not mention to write this thesis. Their love is the most precious things in my life.

Finally, I am grateful to my friend, Yashuang Wang. She is always be there to comfort me whenever I need.

Table of Contents

List of Tables

List of Figures

List of Appendix

Abbreviations

GRAP	Group Role Assignment Problem
RBC	Role-Based Collaboration
GAP	Generalized Assignment Problem
AP	Assignment Problem
IAP	Incremental Assignment Problem
IAA	Incremental Assignment Algorithm
IIAA	Improved Incremental Assignment Algorithm
IGRAP	Improved Group Role Assignment Problem

Chapter 1

1 Assignment Problem

This chapter is a review of the Assignment Problem (AP) and its algorithms. This chapter includes:

- The introduction of AP
- Perfect matching
- Bipartite matching algorithm
- Linear Assignment Problem
- Other Types of Assignment Problem

1.1 Background

In recent years, there are many situations concerning assignment arise in many fields, such as, medical institution, business, transportation, education fields, and sports. The assignment problem is under optimization or operations research branches. It is a widely studied topic in combinatorial optimization problems [1]. In addition, the assignment problem is an important subject for solving many problems around the world [2].

Assignment problems are to assign n items to n other items [3] in an optimal way. The two components are the objective function and the assignments. The objective function reflects the expectation of optimizing as much as possible, while the assignment represents the underlying combinatorial structure. The problem is to minimize the total costs of operating the tasks or maximize the total profit of allocation. A table or matrix can be shown as assignment problem. Generally, the rows represent people or objects to assign, the columns stand for the tasks or things to be assigned [1].

A bipartite graph can be used to describe assignments. The definition of the bipartite graph is: A graph $G = (U, V; E)$ has two vertex sets U and V which are not disjoint. E is an edge set. If every edge connects a vertex of V and there are no edges which have both endpoints in U and V, then G is called bipartite. A subset of the edges such that every vertex of G meets at most one edge of the matching is called a matching M in G. Suppose that the number of vertices in U and V equals n, i.e., $|U| = |V| = n$. If in this case, the matching M is called a perfect matching when each vertex of G coincides with an edge of the matching M. Obviously, every assignment can be shown as a perfect matching [4], shown in Figure 1.

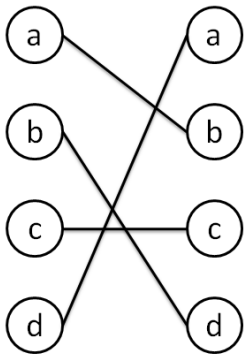


Figure 1: Representation of assignment

Given a bipartite graph $G = (U, V; E)$ and its edge set E and two vertex sets U and V which are not disjoined. The assignment problem is to find a set of n edges in the bipartite graph such that every vertex coincides with exactly one edge. It is also known as the Bipartite Perfect Matching Problem.

1.2 Perfect Matching

We will discuss whether there exists an assignment (i.e., a perfect matching) in a given bipartite graph or not in this section. Hall's Marriage Theorem gives a basic answer to this question in 1935 [5]. There is a necessary and sufficient condition is stated which is known as the Hall's Marriage Theorem for finding the perfect matching in a bipartite graph. For a vertex i/U, let $N(i)$ denote the set of its neighbors, i.e., the set of all vertices j/V which are connected with i by an edge in E. When we consider the vertices in U as young men and the vertices in V as young ladies, the set $N(i)$ contains the friends of i. Moreover, for any subset U of U let $N(U) = \bigcup_{i \in U} N(i)$. The Theorem 1.2.1, 1.2.2 and 1.2.3 are presented by Hall [5].

Theorem 1.2.1. (Hall [5]F, 1935.) *Let $G = (U, V; E)$ be a bipartite graph. It is possible to match every vertex of U with a vertex of V if and only if for all subsets U of U*

$$|U| \leq |N(U)| \text{ (Hall's condition)}$$

Theorem 1.2.2.(Marriage theorem.) *Let $G = (U, V; E)$ be a bipartite graph with $|U| = |V|$. There exists a perfect matching (marriage) in G if and only if G fulfills Hall's condition.*

Theorem 1.2.3.(König's matching theorem [19], 1931.) In a bipartite graph the minimum number of vertices in a vertex cover equals the maximum cardinality of a matching:

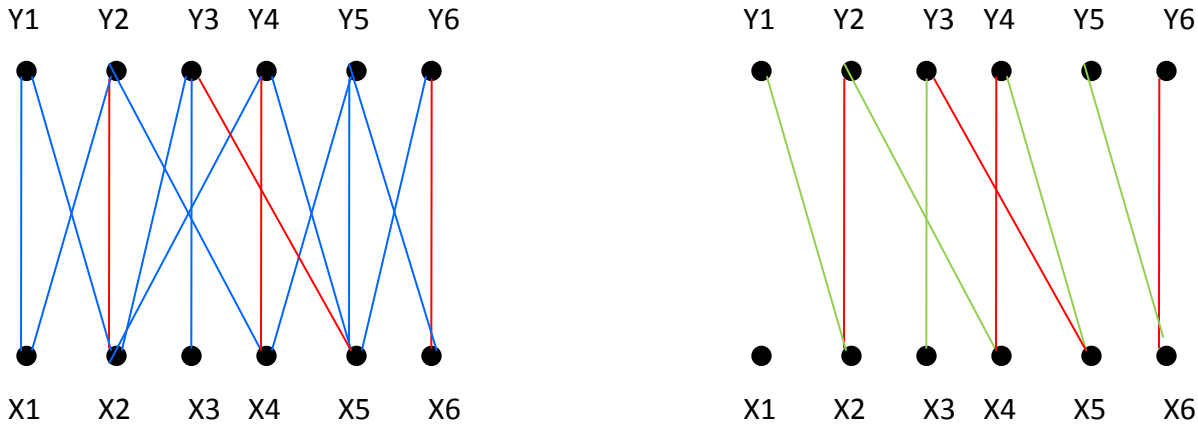
$$\min_{C \text{ vertex cover}} |C| = \max_{M \text{ matching}} |M| .$$

1.3 Bipartite Matching Algorithms

In assignment problems, it is an important subject to finding a maximum matching in a bipartite graph.

1.3.1 A labeling (dual variables) method to find a maximum matching

Given a bipartite graph $G = (U, V; E)$ and a matching M (M might even be empty). If an edge of E belongs to M, it is called a matching edge, if not a non-matching edge. Edges are alternately matching and non-matching is called an Alternating Path. An Augmenting Path P is an Alternating Path that starts from a free (unmatched) vertex and ends on a free (unmatched) vertex.



bipartite graph alternating/ augmenting path

Figure 2: Bipartite graph & alternating/ augmenting graph

In Figure 2, let M be a matching of G. Vertex v is matched if it is the endpoint of an edge in M; otherwise v is free. $Y_2, Y_3, Y_4, Y_6, X_2, X_4, X_5, X_6$ are matched, other vertices are free. Y_5, X_6, Y_6 is an alternating path. $Y_1, X_2, Y_2, X_4, Y_4, X_5, Y_3, X_3$ is an augmenting path.

The definition of augmentation: Let P be an augmenting path with respect to the matching M. There are two rules that the matching augmented by P is obtained [4].

The first rule is the unmatched and matching edges in P change their roles (all previously matching edges of $M \cap P$ become unmatched edges, at the same time, all previously unmatched edges of $M \cap P$ now become matching). The second rule is all matching edges of M which do not lie on the path P remain to be the matching edges.

If M is not a maximum matching in G, then there exists an augmenting path P with respect to M, and $M = M \in P$ is a matching in G with $|M| = |M| + 1$ [7]. The cardinality matching algorithm is presented below [4]:

Cardinality matching algorithm:

M is a matching in graph $G = (U, V; E)$ (possibly $M = \emptyset$);

L contain all unmatched vertices of U;

Labeled vertices on the right side are collected in the set R;

$R := \emptyset$;

while $L \cup R \neq \emptyset$ **do**

choose a vertex x from $L \cup R$;

if x/L **then** Scan_leftvertex(x) **else** Scan_rightvertex(x)

endwhile

Procedure Scan_leftvertex(x)

$L := L \setminus \{x\}$;

while there exists an edge $[x, j]$ with j unlabeled **do**

label j as $l(j) := x$;

$R := R \cup \{j\}$

endwhile

Procedure Scan_rightvertex(x)

$R := R \setminus \{x\}$;

if there is a matching edge$[i, x]$ **then**

label i as $r(i) := x$;

$L := L \cup \{i\}$

else [**comment:** augmentation of the matching]

starting from x, find the alternating path P by backtracking the labels;

$P := \big(\ldots, r\big(l(x)\big), l(x), x\big)$;

$M := M \in P$;

let L contain all unmatched vertices of U;

$R := \emptyset$;

cancel all labels

endif

The algorithm starts with an matching M which is arbitrary. The matching M gradually increases (augments) this matching step-by-step by way of augmenting paths until a maximum matching is reached. At each augmentation, one additional vertex of U is matched. Therefore, there will be at most n augmentations. In addition, every vertex is labeled no more than once for

each augmentation. Therefore, finding a augmenting path requires at most $O(m)$ steps, the operation time is $O(nm)$.

1.3.2 Other Algorithms

Hopcroft and Karp [6] propose an algorithm that not only augments the augmentation by augmenting path, but also augments the matching by the largest disjoint aurmenting path system with the same minimum length. It runs in $O(m\sqrt{n})$ time.

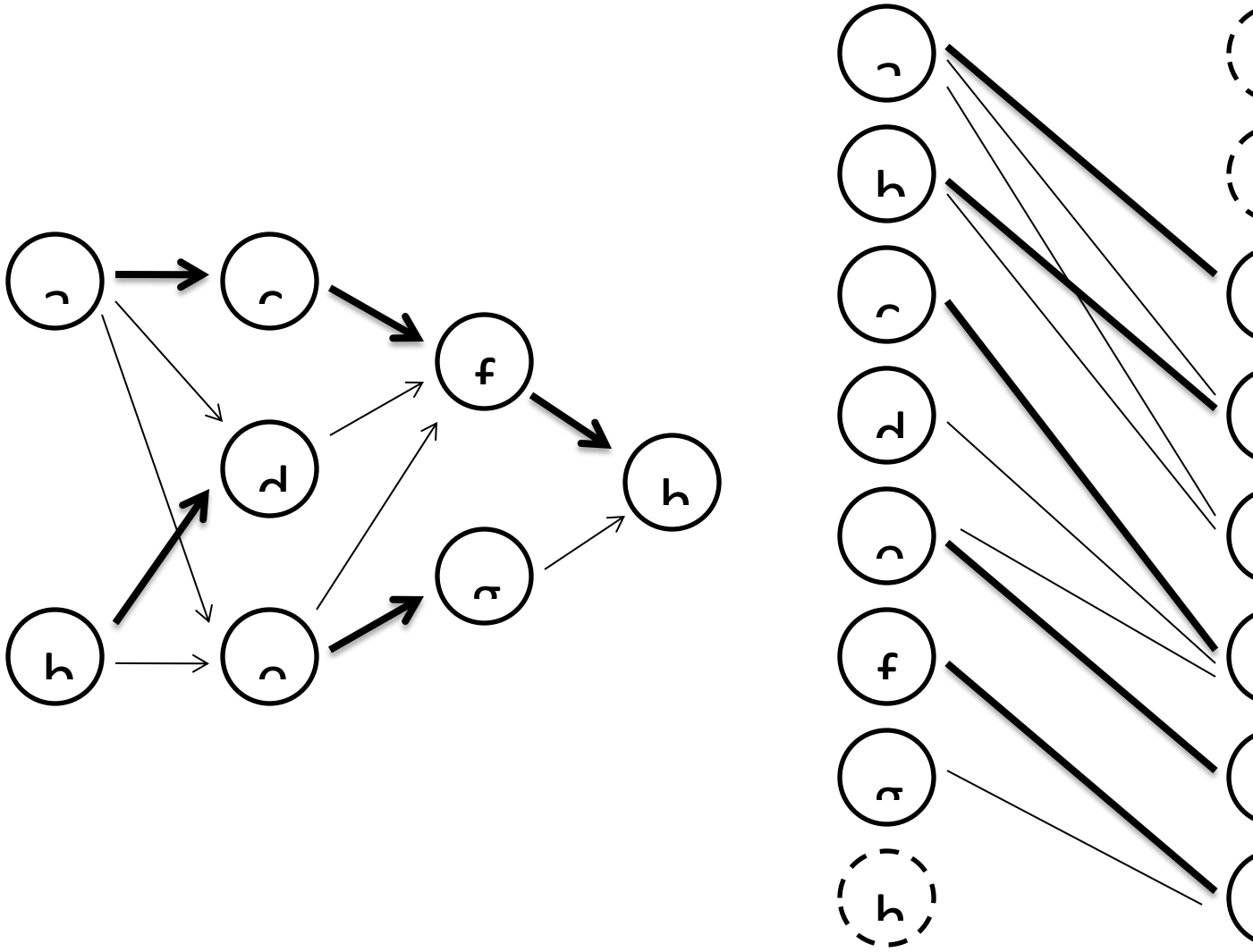
Alt, Blum, Mehlhorn, and Paul [8] do some improvement on Hopcroft and Karp's method for the case of "dense" graphs by using a fast adjacency matrix scanning technique of Cheriyan, Hagerup and Mehlhorn [9]. Therefore they get an algorithm that improves the running time in $O(n^{1.5}\sqrt{m/\log n})$ to find the maximum bipartite matching, where $n = |U| + |V|$, where U and V are vertex sets.

The rank of matrices has a close relationship with maximum matching. Another stochastic method is based on the algorithm which is called fast matrix multiplication and gives complexity of $O(n^{2.376})$ [10]. It is theoretically better for sufficiently dense graphs, but the algorithm is slower in practice.

1.3.3 Applications of the Maximum Matching Algorithm

(1) Vehicle scheduling problems

When planning the operative public transport, the vehicle scheduling problem (VSP) is one of the most important tasks [11]. The set of schedule trips is given that the travel (departure and arrival) time is fixed, as well as the start and end locations, and travel times between all pairs of end stations, the aim is to find the assignment of trips to the vehicles in order to accurately cover each trip once, each vehicle performs a feasible itinerary sequence of trips [12].



(a) (b)

Figure 3: (a) the network which cover all nodes with a minimum number of node disjoint paths (the bold line is shown the network); (b) shows the corresponding maximum matching [4].

A disjoint path problem modeled as this problem in a network. An example is shown in Figure 3(a). Then convert it to a maximum matching problem in a bipartite graph. Figure 3(b) shows the connect lines between the node disjoint paths of Figure 3(a) and matching.

(2) Time slot assignment problem

When using satellites in telecommunication systems, the data first to be remitted are first buffered in the ground station, then the data are transmitted to the satellite in very short data bursts, they are amplified and send back to Earth. The Time Division Multiple Access (TDMA)

technique can be used. A transponder connects the receiving station to the sending station. The time slot assignment problem solves the problem of which switching modes should be applied. It also figures out how long each of them lasts in a given amount of data can be remitted in the shortest possible time [13]. The time complexity is $O(n^4)$.

1.4 Linear Assignment Problem

The general purpose of the assignment problem is to optimize the resources distribution. Resources demand points, and both resources and demand point have the same number [2]. Mathematically, the Linear Assignment Problem can be proposed following:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.4.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,...,n \tag{1.4.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,...,n \tag{1.4.3}$$

$$x_{ij} = 0 \text{ or } 1,\ i = 1,\ldots,n,\ j = 1,\ldots,n \tag{1.4.4}$$

Where c_{ij} is the cost of effectiveness when assigning ith resource to jth demand, x_{ij} is 0 or 1 (as presented in (1.4.4)), and n is the number of resources or demands. The constraints of the assignment problem are defined as (1.4.2) - (1.4.4). Equation (1.4.22) indicates that each resource i only can be assigned to one demand j, while (1.4.3) shows that each demand j only can be assigned to one resource i.

However, this theorem is not directly an efficient method for finding a perfect matching. In the early days of mathematical programming, labeling methods were used to create a perfect matching of the $O(n^3)$ complexity. Several authors improved on these methods later. One of the well-known methods is Hopcroft and Karp [6], which show that a perfect matching can be found in $O(n^{5/2})$ times. We will present this algorithm below.

Regarding every assignment problem, there is a matrix called cost or validity matrix $[c_{ij}]$, where $c_{ij}$is the assigning cost of ith resource to jth demand. In this paper, it is called an

assignment matrix, where each resource can only be assigned to a requirement and represent it, as given in the following:

$$\begin{pmatrix} c_{11}c_{12}c_{13}\cdots c_{1n} \\ c_{21}c_{22}c_{23}\cdots c_{2n} \\ \vdots \ \vdots \ \vdots \ \vdots \ \vdots \\ c_{n1}c_{n2}c_{n3}\cdots c_{nn} \end{pmatrix} \tag{1.4.5}$$

Which is always a square matrix.

Here is a real world scenario. Assume that n jobs are to be assigned to n machines (or workers) in the best possible way. Let us assume that machine j needs c_{ij} time units to process job i. We want to minimize the total completion time. If we assume that the machines work in series, we must minimize the linear sum objective function. If we assume that the machines work in parallel, we have to minimize the bottleneck objective function.

This example shows different objective functions of interest. When it is necessary to minimize cost, the sum objective is usually used. If a time need to be minimized, a so-called bottleneck objective function is often used. Although this function is not written in linear form, the optimization problem with this objective function is called "linear" compared to the quadratic problems introduced in Section 1.5.

1.4.1 Linear Sum Assignment Problem

The most well-known problems in linear programming and in combinatorial optimization is the linear sum assignment problem (LSAP). Informally, we are given an $n \times n$ cost matrix $C = (c_{ij})$ and we want to match each row to a different column in order to minimize the sum of the corresponding entries. In other words, we want to select n elements of C so that there is only one element in each row and one in each column and the sum of the corresponding costs is a minimum.

Alternatively, LSAP can be defined through a graph theory model, for example, a bipartite graph $G = (U, V; E)$ having a vertex of U for each row, a vertex of V for each column, and cost c_{ij} associated with edge $[i, j](i, j = 1,2, \ldots n)$. The problem is then to determine the minimum cost perfect matching in G (weighted bipartite matching problem: find a subset of

edges so that each vertex happens to belongs to exactly one edge and the sum of the costs of these edges is a minimum).

They mainly occur in sub-problems in more complex cases, such as the travelling salesman problems, vehicle routing problems, personnel assignments and similar problems in practice [3]. Neng [14] describes an interesting application in railway systems. He considers the problem of assigning engines to trains due to traffic restrictions and expressed this problem as a linear assignment problem.

A large number of sequential and parallel algorithms has been developed for the LSAP, such as primal-dual algorithms, simplex-like methods, cost operation algorithms, forest algorithms and relaxation approaches. For a survey on these methods and available computer programs see the recent article of Burkard and Çela [15] or the annotated bibliography of Dell' Amico and Martello [16]. Nowadays it is possible to solve large scale dense LSAPs (with $n \approx 10^6$) within a couple of minutes, see [17].

Although $O(n^3)$ is the best worst case complexity for sequential linear sum assignment algorithms, Karp [18] develope an algorithm with the expected running time of $O(n^2 \log n)$ in the case of independent and uniformly distributed cost coefficients c_{ij} in [0, 1].This algorithm is a special implementation of the classical shortest augmenting path algorithm. It uses priority queues to compute shortest augmenting paths in $O(n^2 \log n)$ time which produces a worst case time complexity of $O(n^3 \log n)$.

1.4.1.1 Mathematical Model

The linear sum assignment problem (LSAP) can be stated as

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1.4.1.1.1)$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ldots,n$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ldots,n$$

$$x_{ij} = 0 \text{ or } 1,\ i = 1,\ldots,n,\ j = 1,\ldots,n$$

By associating dual variables u_i and v_j with assignment constraints (1.4.1.1.2) and (1.4.1.1.3), respectively, the dual problem is

$$\max \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \qquad (1.4.1.1.2)$$

$$\text{s.t. } u_i + v_j \leq c_{ij}\ (i, j = 1,2, \ldots n) \qquad (1.4.1.1.3)$$

By duality theory, a pair of solutions respectively feasible for the primal and the dual is optimal if and only if

$$x_{ij}\left(c_{ij} - u_i - v_j\right) = 0\ (i, j = 1,2, \ldots n) \qquad (1.4.1.1.4)$$

1.4.1.2 An $O(n^4)$ implementation of the Hungarian Algorithm

The Hungarian algorithm is considered to be a predecessor of the primal-dual method for linear programming. It begins with a feasible dual solution u, v satisfying (1.4.1.1.3) and a partial primal solution (in which less than n rows are assigned) satisfying the condition (1.4.1.1.4) with respect to u, v. Each iteration solves a restricted primal problem independent of the costs, attempting to increase the cardinality of the current assignment by operating on the partial graph of $G = (U, V; E)$ that only contains the edges of E having zero reduced costs. If the attempt is successful, a new primal solution in which one more row is assigned is obtained. Otherwise, updating the current dual solution to get a new edge with zero reduction.

The main idea of the method is as follows: consider we only use the edge of weight 0 (called the "0-weight edges") to find the perfect matching. Obviously, these edges will be the solution of the assignment problem. If we cannot find perfect matching on the current step, then the Hungarian algorithm changes weights of the available edges so that the new 0-weight edges appear and these changes do not affect the optimal solution [38].

Preliminaries. For each vertex from the left part, find the minimal outgoing edge and subtract its weight from all weights of the connection to the vertex. This will introduce 0-weight edges (at least one). Apply the same procedure for the vertices in the right part.

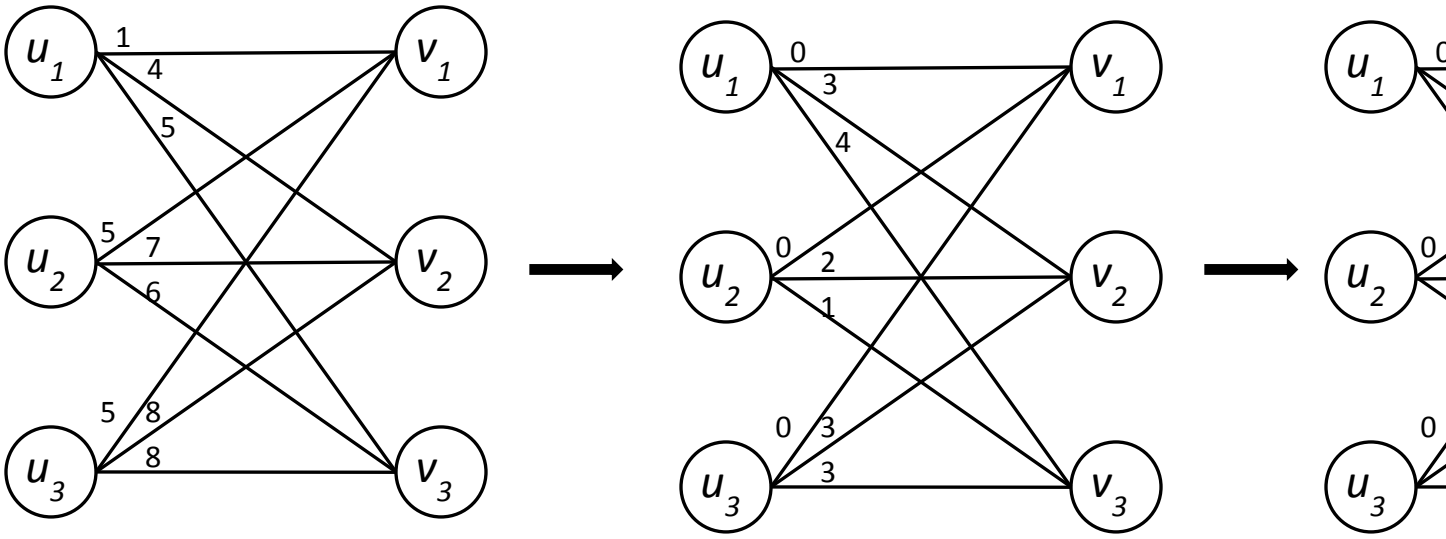
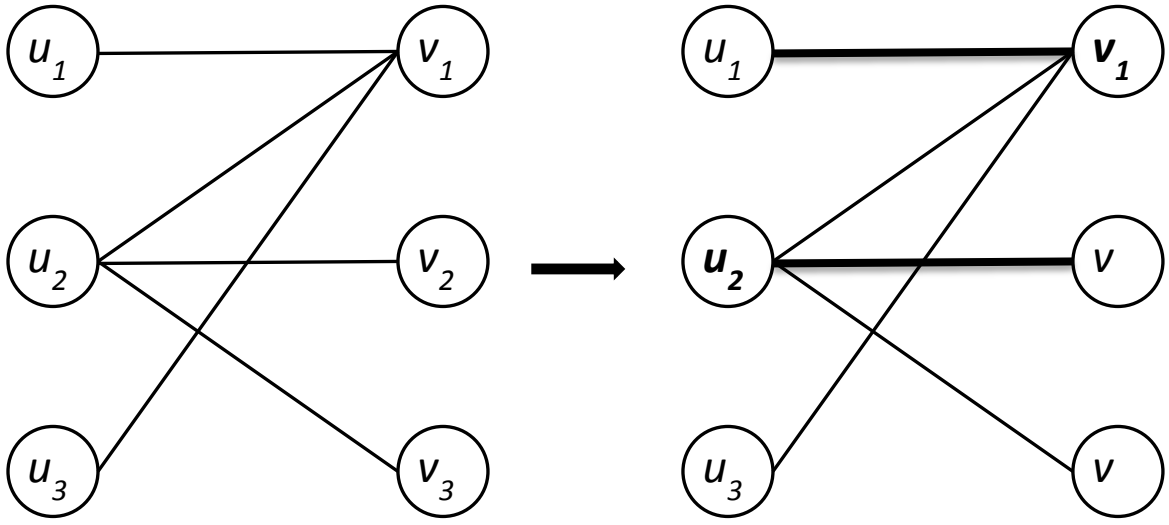


Figure 4: The preliminaries [38]

Step 1. Find the maximum matching using only 0-weight edges (using augmenting path algorithm, etc.). If it is perfect, then the problem is solved. Otherwise find the minimum vertex cover E (for the subgraph with 0-weight edges only), the best way to do this is to use Köning's matching theorem.

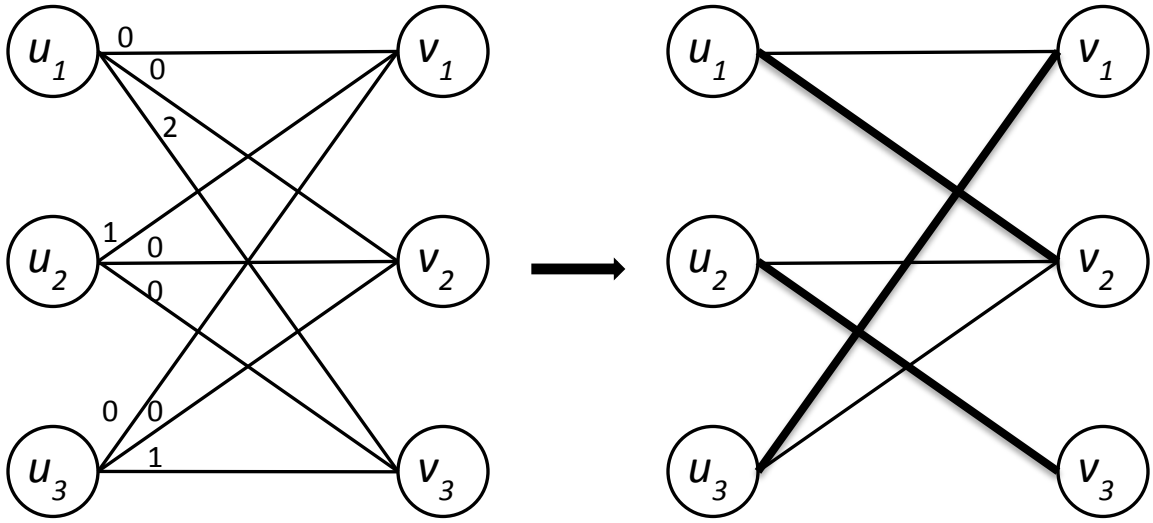


(a) (b)

Figure 5: (a) Graph with 0-weight edges only; (b) Maximum matching and minimum vertex cover [38]

Step 2. Let $\Delta = min_{i \notin U, j \notin V}(c_{ij})$ and adjust the weights by using the rule as follows:

$$c_{ij} = \begin{cases} c_{ij} - \Delta, i \notin U \wedge j \notin V \\ c_{ij}, \quad i \in U \vee j \in V \\ c_{ij} + \Delta, i \in U \wedge j \in V \end{cases}$$



(a) (b)

Figure 6: (a) Graph with modified weights (delta=1); (b) Minimum matching [38].

Step 3. Repeat **Step 1** until solved.

In this example, we are using the Hungarian Algorithm to solve the minimum value of the matrix. The perfect matching is $(u_1, v_2), (u_2, v_3), (u_3, v_1)$ which is 4+6+5=15.

But there is a nuance here, finding the maximum matching in Step 1 on each iteration will cause the algorithm to become $O(n^5)$. In order to avoid this situation, we can just modify the matching of the previous step at each step, which only takes $O(n^2)$ time. It's easy to see that since at least one edge becomes 0-wieght at a time, n^2 iterations will occur. Therefore, the overall complexity is $O(n^4)$.

1.4.1.3 An $O(n^3)$ implementation of the Hungarian Algorithm

We will introduce the maximum-weighted matching problem in this section. Obviously, it is easy to transform minimum problem to the maximum one, just by setting:

$$w(u,v) = -w(u,v), \forall (u,v) \in E$$

$$\text{or } w(u,v) = M - w(u,v), M = \max_{(u,v)\in E} w(u,v)$$

Given a matching M, a path is **alternating** that begins with an unmatched vertex and whose edges belong alternately to the matching and not to the matching. It is called **an alternating path**. **An augmenting path** is an alternating path that starts from and ends on free (unmatched) vertices. All alternating paths originating from a given unmatched node form **a Hungarian tree**.

The algorithm assigns dual variables α_i and β_j to each node U and node V respectively. The assignment problem is feasible when $\alpha_i + \beta_j \geq W_{ij}$. The Hungarian algorithm maintains feasible values for all the α_i and β_j from initialization to termination. When $\alpha_i + \beta_j = W_{ij}$, an edge in the bipartite graph is called admissible. The sub-graph containing only the currently admissible edges is called the equality sub-graph G'.

Starting from an empty matching, the basic strategy adopted by the Hungarian algorithm is to repeatedly search for augmenting paths in the equality sub-graph. If an augmenting path is found, the current match set is augmented by flipping the matched and unmatched edges along this path. Because there is one more unmatched than matched edge, this flipping increases the cardinality of the matching by one, completing a single stage of the algorithm. If an augmenting path is not found, adjust the dual variables to bring extra edges into the equality sub-graph by making them admissible and the continue searching. These stages of the algorithm are executed to determine n matches, at which point the algorithm terminates [19].

An outline of the Hungarian algorithm for the assignment problem is shown below [19].

Hungarian Algorithm:

Input: A bipartite graph, $\{U, V; E\}$ (where $|U| = |V| = n$) and an $n \times n$ matrix of edge

---

weights W_{ij}

Output: A complete matching, M

1. Perform initialization:
   (a) Begin with an empty matching, $M = \emptyset$
   (b) Assign feasible values to the dual variables α_i and β_j as follows:

$$\forall u_i \in U, \alpha_i = 0 \quad (1.4.1.3.1)$$

$$\forall v_j \in V, \beta_j = max_i\, W_{ij} \quad (1.4.1.3.2)$$

2. Perform n stages of the algorithm, each given by the routine **Stage**.
3. Output the matching after the n^{th} stage: $M = M_n$.

Stage:

1. Designate each exposed (unmatched) node in U as the root of a Hungarian tree.

2. Grow the Hungarian tree rooted at the exposed nodes in the equality sub-graph. Designate the indices i of nodes u_i encountered in the Hungarian tree by the set I^*, and the indices j of nodes v_j encountered in the Hungarian tee by the set J^*. If an augmenting path is found, go to **Step 4**. If not, and the Hungarian trees cannot be grown further, proceed to **Step 3**.

3. Modify the dual variables α_i and β_j as follows to add new edges to the equality subgraph. Then go to **Step 2** to continue the search for the augmenting path.

$$\theta = \min_{i \in I^*, j \notin J^*} (\alpha_i + \beta_j - W_{ij})$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i - \theta & i \in I^* \\ \alpha_i & i \notin I^* \end{cases}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J^* \\ \beta_j & j \notin J^* \end{cases}$$

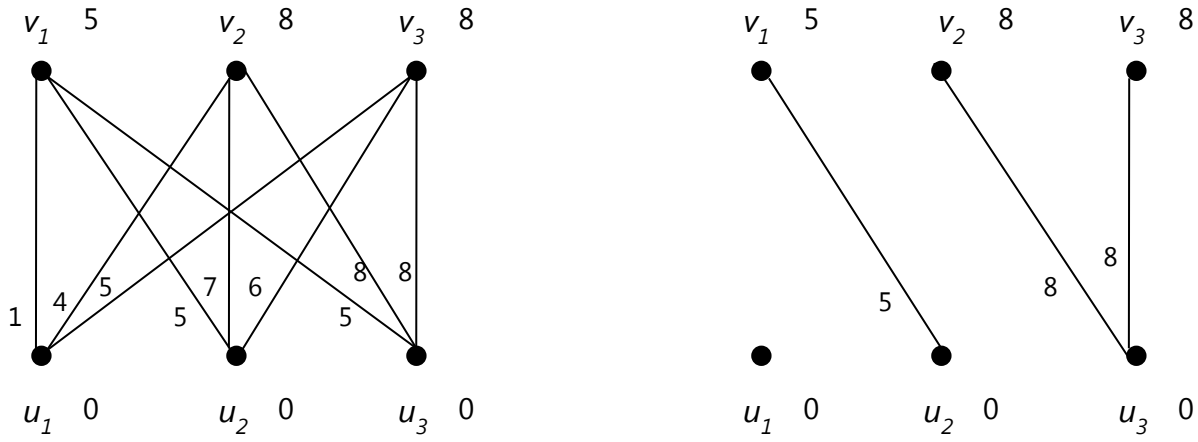
4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path. That is, M_k (the new matching at stage k) is given by $(M_{k-1} - P) \cup (P - M_{k-1})$, where M_{k-1} is the matching from the previous stage and P is the set of edges on the selected augmenting path.

Example: Consider the 3 × 3 weighted bipartite graph described by its weight matrix in the following:

	v_1	v_2	v_3
u_1	1	4	5
u_2	5	7	6
u_3	5	8	8

Figure 7: **3 × 3** matrix

Initialing the graph, trivial labeling and associated equality graph.



(a) (b)

Figure 8: (a) Original graph; (b) Equality subgraph+Matching.

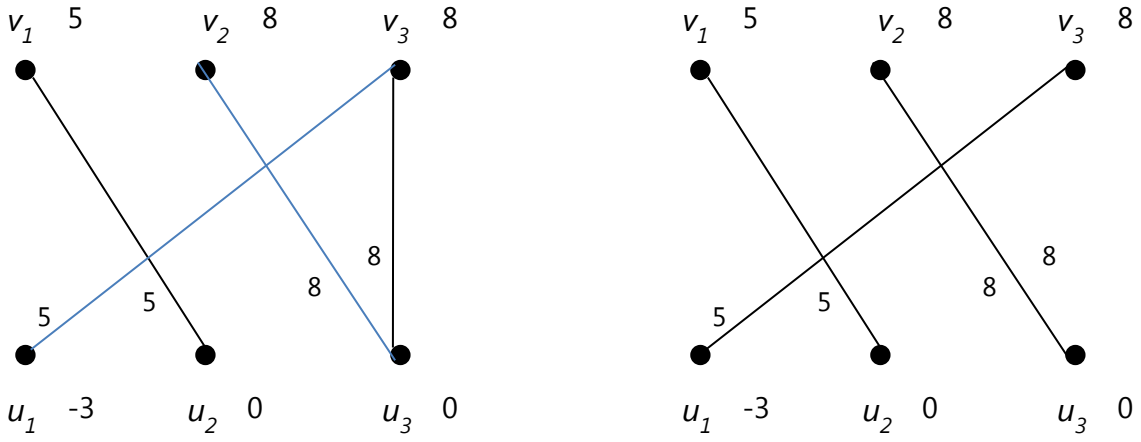
The Hungarian tree can be grown rooted at the nodes u_1 in the equality subgraph. $I^* = \{u_1\}$ and $J^* = \emptyset$. $M = \{(u_2, v_1), (u_3, v_2), (u_3, v_3)\}$. The augmenting path cannot be found.

$$\theta = min_{i \in I^*, j \notin J^*} \begin{cases} 0 + 5 - 1 \ (u_1, v_1) \\ 0 + 8 - 4 \ (u_1, v_2) \\ 0 + 8 - 5 \ (u_1, v_3) \end{cases} = 3$$

Reduce labels of I^* by 3. $I^* = \{u_1\}$ and $J^* = \{v_3\}$. Then we got the new equality subgraph and the augmenting path u_1, v_3, u_3, v_2. Flipping matched to augment the current

matching and unmatched edges along the selected augmenting path.

$M_k = \{(u_2, v_1), (u_3, v_2), (u_1, v_3)\}$.



(a) (b)

Figure 9: (a) New equality subgraph; (b) Matching.

In each iteration, we increment matching. Therefore, we have n iterations [4]. At each iteration, each edge of the graph is used no more than once when finding augmenting path, so we've got $O(n^2)$ complexity. Concerning labeling we update slack array each time when we insert vertex from U into I^*, so each iteration does not occur more than n times, updating slack takes $O(n)$ operations, so again we've got $O(n^2)$. Updating labels occur no more than n times per iterations (since we add at least one vertex from V to J^* on each iteration), it takes $O(n)$ operations. Therefore the total complexity of this implementation is $O(n^3)$.

Jonker and Volgenant [20] develope an improved $O(n^3)$ Hungarian algorithm. Fortran implementations of Hungarian algorithm are proposed by McGinnis [21], Carpaneto and Toth [22], and Carpaneto, Martello, and Toth [23]. The Carpaneto and Toth [22] paper, which includes the Fortran listing of their code, provides computational comparisons with the primal simplex algorithm by Barr, Glover, and Klingman [24].

1.4.1.4 Other algorithms

Dinic and Kronrod [25] has proposed a different approach that is completely independent of the linear programming duality theory.

The most efficient LSAP algorithm is based on shortest augmenting path techniques. In the early 1960s, Hoffman and Markowitz [26] observe that an LSAP can be solved through a sequence of n shortest paths on cost matrices of increasing size from 1×1 to $n \times n$. However such matrices can include negative costs, so each shortest path search would require $O(n^3)$ time. In early 1970s Tomizawa [27] and Edmonds and Karp [28] study shortest path algorithms for the min-cost flow problem observe that by using the reduced costs, Dijkstra algorithm can be applied to obtain an $O(n^3)$ time algorithm for LSAP.

Dantzig [29] specialize the primal simplex algorithm into a network problem, which is the starting point for all primal simplex-based algorithms for LSAP. Gavish, Schweitzer and Shlifer [30] give computational results on the effect of various pivoting rules on the number of degenerate pivots in the solution of LSAP. In general, the primal algorithms are less efficient than other methods.

1.4.2 The Linear Bottleneck Assignment Problem

Fulkerson, Gllicksberg, and Gross [31] introduce the problem of linear bottleneck assignment. It happens when a job is assigned to a parallel machine to minimize the latest completion time. Another application is to locate objects in space. Let n jobs and n machines be given. The cost coefficient c_{ij} is the time required for machine j to complete job i. If the machines work in parallel and we want to assign the jobs to the machines such that the latest completion time is as early as possible, it can mathematically be written as

$$\min \max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1.4.2.1)$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ldots,n$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ldots,n$$

$$x_{ij} = 0 \text{ or } 1,\ i = 1,\ldots,n,\ j = 1,\ldots,n$$

Solving an LBAP with cost matrix $C = (c_{ij})$ can produce very good results in practice. A similar technique can be used to track missiles in space. If their locations at two different times t_1 and t_2 are known, the squared Euclidean distances between any pair of old and new locations are calculated and the corresponding linear bottleneck assignment problem is solved to match the points in the right way.

1.4.3 Other Types of Linear Assignment Problems

1. Algebraic Assignment Problem

Sum and bottleneck assignment problems can be viewed as special situations of a more general model - the algebraic assignment problem which is introduced by Burkard, Hahn, and Zimmermann [32]. It allows people to develop and solve linear assignment problems within a general framework.

2. Sum-k Assignment Problem

Given an $n \times n$ cost matrix $C = (c_{ij})$ and a value k is not greater than n, the sum-k assignment problem is to assign each row to a different column such that the sum of the k largest selected costs is a minimum. Grygiel [33] designs an $O(n^5)$ algorithm with real coefficients.

3. Balanced Assignment Problem

Martello, Pulleyblank, Toth, and de Werra [34] introduce the balanced assignment problem in a more general framework for balancing optimization problems, minimizing the spread of an assignment solution.

1.5 Other Types of Assignment Problem

1.5.1 Quadratic Assignment problems

The quadratic assignment problem (QAP) is introduced by Koopmans and Beckmann [35] in 1957 as a mathematical model for the location of indivisible economical activities. A set of n facilities has to be allocated to a set of n locations. We give three $n \times n$ input matrices:

$A = (a_{ik})$, $B = (b_{jl})$, and $C = (c_{ij})$, where a_{ik} is the flow between facility i and facility k, b_{jl} is the distance between location j and location l, and c_{ij} is the cost of placing facility i at location j. We assume that the total cost depends on the flow between facilities multiplied by their distance and the cost for placing a facility at a certain site. The goal is to assign each facility to a location to minimize the total cost.

The quadratic assignment problem can be modeled as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{ik} b_{jl} x_{ij} x_{kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1.5.1.1)$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,...,n$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,...,n$$

$$x_{ij} = 0 \text{ or } 1,\ i = 1,\ldots,n,\ j = 1,\ldots,n$$

Quadratic assignment problems are so-called NP-hard problems. This means that an optimal solution can only be found by enumeration of all possibilities unless $P = NP$.

1.5.2 Multi-index Assignment problems

In 1968, Pierskalla [36] introduces Multi-index assignment problems as a natural extension of linear assignment problems. For a long time only 3-index assignment problems have been considered, and in recent years, more than 3 indices problems have been investigated, mainly in the context of multi-target tracking and data association problems [37].

Chapter 2

2 Group Role Assignment Problem

This chapter is a review of the Group Role Assignment Problem (GRAP). Generalized Assignment Problem (GAP) is related to GRAP, this chapter presents:

- The introduction of GRAP,
- The introduction of GAP,
- The mathematical formulation of the GAP,
- Literature review of the GAP,
- Role-Based Collaboration,
- The mathematical formulation of the GRAP, and
- The solution of GRAP

2.1 Introduction

In the real world, different people can be involved in different roles in different fields, such as sports players, doctors, teachers, etc. Roles are commonly concepts in many fields, e.g., behavioral science, sociology, drama, social psychology, management and psychology [39]. Therefore, collaboration is necessary to get the optimal performance of the entire system. Role-Based Collaboration (RBC) is a method to promote an organizational structure, to provide ordered system behavior, and to integrate system security for both human and nonhuman entities that collaborate and coordinate their activities within systems [40]. Take a soccer team as an example; every football player has different performance when assigned to each role. How to assign football players to different roles and get their best performances in that role is the ultimate purpose for the coach. In order to solve these kinds of problems, RBC is a useful and functional methodology.

In RBC, role assignment is a crucial task that affects the collaboration efficiency and the level of satisfaction of all the participating members involved in. It can be divided into three steps: agent evaluation, group role assignment, and role transfer [40]. Group role assignment problem (GRAP) initiates a group by assigning roles to its members or agents to achieve its highest performance. Considering the same example above, in a soccer team, if a coach wants to pick up 11 football players from 20 players for four roles: one goalkeeper, four backs, three midfields, and three forwards, how to make role assignment to optimize the whole team's performance is a typical GRAP.

GRAP can be transferred to the Generalized Assignment Problem (GAP). The well-known Kuhn-Munkres (K-M) algorithm is designed to solve the GAP with the complexity of $O(n^3)$, GRAP can be solved efficiently.

The objective of GAP is to find an assignment in which all agents minimize their costs or the total profit of the assignment is maximized. The GAP has been given the optimal or approximate solutions by different algorithms.

2.2 Generalized Assignment Problem

The generalized assignment problem (GAP) is a problem in combinatorial optimization. When the number of tasks and agents are equal, it is known as a generalization form of a classical Assignment Problem (AP). This means that for GAP, the number of agents assigned to each task could be different.

The goal of GAP is to find an assignment in which all agents minimize their costs or maximize the total profit. It has been applied to many applications, such as various routing problems and flexible manufacturing systems [41].

2.2.1 The Mathematical Formulation of the GAP

Given n jobs ($j = 1, \dots n$) and m agents ($i = 1, \dots, m$), each job should be assigned to only one agent to maximize the total profit without exceeding their budget.

GAP can be formulated as an integer programming problem

$$\begin{cases} max \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} & (2.2.1) \\ \qquad\qquad s.t. & \\ \sum_{j=1}^{n} w_{ij}x_{ij} \le c_i \quad i = 1, \dots, m; & (2.2.2) \\ \sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \dots, n; & (2.2.3) \\ x_{ij} \in \{0,1\} \qquad i = 1, \dots, m, j = 1, \dots, n; & (2.2.4) \end{cases}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to agent } i \\ 0 & \text{otherwize} \end{cases}$$

The parameters can be defined as follows: the profit of assigning job j to agent i is represented by p_{ij}, the weight of assigning job j to agent i is represented by w_{ij}, the budget allocated for agent i is denoted by c_i. The objective function is to maximize the total profit of all the assignments. Constraint (2.2.2) is the limit of budget. Constraint (2.2.3) ensures every agent

is assigned exactly one job. Constraint (2.2.4) outlines the decision valuable and specify the ranges of both variables i and j.

The minimization version of the problem can also be encountered in the literature: by defining c_{ij} as the cost required to assign item j to task i. The formula is

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (2.2.5)$$

subject to (2.2.2), (2.2.3), (2.2.4)

GAP is a generalization of the Assignment Problem (AP). When $w_{ij} = 1$ for all $i \in m, j \in n$ and $m = n$, GAP is reduced to AP. AP has been solved by Hungarian Method (also known as K-M Algorithm) in polynomial time.

When job j assigns to agent i with weight w_j, profit p_j and capacity c_i, the 0-1 Multiple Knapsack Problem is a special case of the GAP. Furthermore, GAP can be interpreted as a specialized Transportation Problem when the quantity demanded at each destination should be supplied y a single origin and $w_{i,j}$ is constant for each i.

2.2.2 Literature review of the GAP

There are several algorithms which have been proposed to obtain a better solution for GAP. These algorithms can be divided into three categories: the branch and bound scheme, the branch and price scheme and the heuristic methods.

- The branch and bound scheme

There are four procedures for the branch and bound: an upper bounding procedure, a lower bounding procedure, a branching strategy and a searching strategy.

Ross and Soland [42] develop the first branch and bound algorithm to solve GAP. They reach the lower bounds by relaxing the capacity constraints. Martello and Toth [43] consider exact algorithms for the zero-one knapsack problem and their average computational performance; the study is extended to the other linear knapsack problems and to approximate

algorithms in Martello and Toth [44]. Fisher, Jalikumar and Wassenhove [45] use heuristic bounds which obtained from a Lagrangian relaxation with multipliers by adjustment methods to obtain the lower bounds in the branch and bound procedure. Guignard and Rusenwein [46] present a new algorithm which is effectively solves problems with up to 500 variables. This algorithm requires fewer enumeration nodes and shorter operation times than existing procedures. Improved performance stems from: an enhanced Lagrangian dual ascent procedure, solving a Lagrangian dual at each enumeration node; adding a surrogate constraint to the Lagrangian relaxed model; and an elaborate branch and bound scheme. Drexl [47] presents a hybrid branch and bound /dynamic programming algorithm with a (rather efficient Monte Carlo type) heuristic upper bounding technique as well as various relaxation procedures for determining lower bounds. Nauss [48] describes a special purpose branch-and-bound algorithm that utilizes linear programming cuts, feasible-solution generators, Lagrangean relaxation, and subgradient optimization. In addition, Nauss [49] presents a special purpose branch and bound algorithm that utilizes linear programming cuts, feasible solution generators, Lagrangean relaxation and subgradient optimization to the elastic generalized assignment problem (EGAP). Posta, Ferland and Michelon [50] propose a simple exact algorithm for solving the GAP. They redefine the optimization problem into a sequence of decision problems, and they applied variable-fixing rules to solve these effectively. The decision problems are solved by variable-fixing rules to prune the search tree.

- The branch and price scheme

The branch and price scheme employs both column generation and branch-and-bound to obtain optimal integer solutions.

Savelsbergh [51] firstly presents branch and price algorithm to solve the GAP. Martello and Toth [52] propose a combination of branch and price algorithm. Nasberg [53] introduces a new approach which is based on a reformulation of GAP into an equivalent problem, which is then relaxed by traditional Lagrangian relaxation techniques. The reformulation is created by introducing a set of auxiliary variables and a number of coupling constraints. By relaxing the coupling constraints, they get subproblems where both types of constraint structures present in the GAP are active. Ceselli and Righini [54] propose a branch and price algorithm for multilevel

generalized assignment problem which is based on a decomposition into a master problem with set-partitioning constraints and a pricing subproblem that is a multiple-choice knapsack problem.

- The heuristic methods

Based on the enumeration strategies, some problems still cannot be solved in reasonable computation time. As a result, many heuristic approaches were designed to find high quality solutions [60].

Cattrysse, Salomon and Van Wassenhove [55] is based on column generation techniques, and yields both upper and lower bounds. A column is represented as a feasible assignment of a subset of tasks to a single agent. The main problem is formulated as a set partitioning problem. New columns that have been obtained will be added to the main problem by solving a knapsack problem for each agent. A dual ascent procedure can be solved using LP relaxation of the set partitioning problem. On a set of relatively hard test problems the heuristic is able to find solutions that are on average within 0.13% from optimality.

Lorena and Narciso [56] propose relaxation heuristics for the problem of maximum profit assignment of GAP. Using Lagrangian or surrogate relaxation, the heuristics perform a subgradient search obtaining feasible solutions. Naricso and Lorena [58] find good feasible solutions by using relaxation multipliers with efficient constructive heuristics.

Haddadi [57] defines a new Lagrangian heuristic for the generalized assignment problem (GAP). The heuristic is based on a Lagrangian decomposition of the problem in which a substitution of variables is performed and the constraints defining the substituted variables are then dualized in a Lagrangian relaxation of the problem. Haddadi and Ouzia [59] describe a new heuristic, applied at each iteration of the SM, which tries to exploit the solution of the relaxed problem, by solving a smaller generalized assignment problem.

Amini and Racer [61] introduce variable depth search heuristic (VDSH) which is used to solve the GAP. VDSH is defined as a generalization of local search in which the size of the neighborhood adaptively changes to traverse a larger search space. Then they [62] develope a hybrid heuristic (HH) around the two heuristics called VDSH and heuristic generalized assignment problem (HGAP). Yagiura, Yamaguchi and Ibaraki [63] propose a heuristic

algorithm based on variable depth search procedure (VDS) for solving the GAP. The main idea is to adaptively change the size of a neighborhood so that it can effectively traverse a larger search space while keeping the amount of computational time reasonable. Yagiura, Yamaguchi and Ibaraki [64] develop a variable depth search (VDS) algorithm. To further improve the performance of the VDS, they examine the effectiveness of incorporating branching search processes to construct the neighborhoods. Lin et al [65] makes further observations on VDSH method through a series of computational experiments. They propose six different strategies for the improvement procedure each of which alternatively creates one action set in each of the loop iterations.

Osman [66] introduces a λ-generation mechanism is introduced. Different search strategies and parameter settings are investigated for the λ-generation descent, hybrid simulated annealing/tabu search and tabu search heuristic methods. Yagiura, Yamaguchi and Ibaraki [67] suggest a tabu search algorithm for the generalized assignment problem, which is one of the representative combinatorial optimization problems known to be NP-hard. The algorithm features an ejection chain approach, which is embedded in a neighborhood construction to create more complex and powerful moves. Dıaz and Fernández [69] create a simple and flexible tabu – search algorithm for solving the GAP. The algorithm uses recent and medium-term memory to dynamically adjust the weight of the penalty incurred for violating feasibility. The algorithm provides good quality solutions in competitive computational times.

Lourenco and Serra [73] present new metaheuristics for the generalized assignment problem based on hybrid approaches. One metaheuristic is a MAX-MIN Ant System (MMAS). The heuristic is combined with local search and tabu search heuristics to improve the search. A greedy randomize adaptive search heuristic (GRASP) is also proposed.

Yagiura et al. [68] propose a new algorithm to prove that this problem is more effective than the previous existing methods. The algorithm uses a path re-linking approach, a mechanism for generating new solutions by combining two or more reference solutions. It also uses an ejection chain approach embedded in a neighborhood construction to create more complex and powerful movements.

Chu and Beasley [70] firstly propose a genetic algorithm to solve the GAP. A fitness-unfitness pair for evaluation function and a heuristic operator helps to increase the cost and feasibility of the solution. This algorithm is used as a heuristic algorithm to help improve the cost and feasibility of GAP. Wilson [71] suggests another algorithm for GAP that operates in a dual sense. This algorithm attempts to genetically restore feasibility to a set of approximate optimal solutions. The new approach is presented by Feltl and Raidl [72] is based on a previously published, successful hybrid genetic algorithm and includes as new features two alternative initialization heuristics, a modified selection and replacement scheme for handling infeasible solutions more appropriately, and a heuristic mutation operator. Lorena, Narciso & Beasley [74] suggest an application of the Constructive Genetic Algorithm (CGA) to the Generalized Assignment Problem (GAP). Compared to a traditional genetic algorithm (GA), CGA presents some new features. When applying CGA to GAP, they consider the GAP to be a clustering problem. A binary representation is used for schemata and structures, and an assignment heuristic allocates items to knapsacks.

2.3 Group Role Assignment Problem (GRAP)

2.3.1 Role-Based Collaboration

Role-based collaboration (RBC) is an emerging computational methodology that uses roles as the primary underlying mechanism to facilitate collaboration activities [75], providing ordered system behavior, and consolidate system security for both human and non-human entities. And collaborate and coordinate with their activities and systems [76]. It consists of a set of concepts, principles, models, and algorithms.

Collaboration is when a task completion needs more than one individual. The research in the fields of collaboration theory, technologies, and systems helps people undertake collaboration in a more efficient and satisfactory manner [75]. Collaboration can be divided into five categories: collaboration between people in the reality; computer-supported cooperative work (CSCW); human-computer/machine interaction (HCI); distributed systems; and a computer system, which are collaboration between system components. Task assignment and coordination are two major aspects of collaboration. RBC is focused on providing better task assignments to

save the effort of coordination, which is considered to be more complex issue than task assignments [75].

The RBC life cycle consists of three major tasks: role negotiation, role assignment and role execution [40]. Obviously, role assignment is an important aspect of RBC. It greatly affects the efficiency of collaboration and the satisfaction of the members involved in the collaboration. Figure 10 shows the life cycle of RBC.

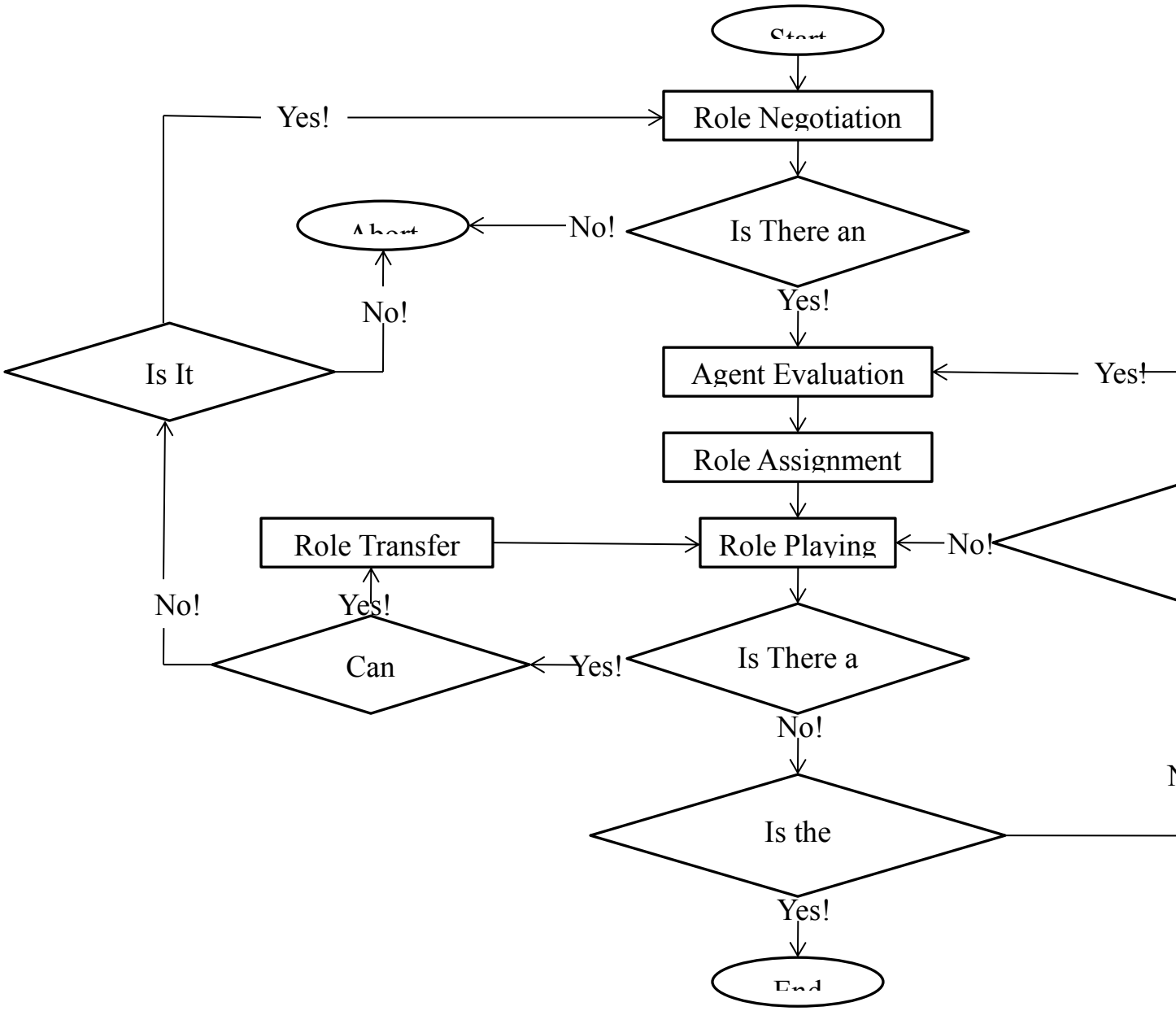


Figure 10： The life cycle of RBC.

After more than a decade continuous effort, RBC-related research has been developed into a discovery method in the field of collaboration-systems research, including the role transfer problem, the group role assignment problem, and so on.

2.3.2 The Mathematical Formulation of the GRAP

GRAP was first proposed by H. Zhu and R. Alkins in 2009 [76]. It aims at finding the maximum total profit among n roles to m agents with the agent evaluation result. The unit profit is represented by $P_{i,j}$. Each role can be assigned to more than one agent and one agent can only receive one role.

The mathematical formulation of GRAP is:

$$\begin{cases} max \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} & (2.3.1) \\ s.t. & \\ \sum_{i=1}^{n} x_{ij} = L[j] \qquad j = 1, \dots, n, & \\ L[j] \in N, \text{N is a set of natural numbers;} & (2.3.2) \\ \sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \dots, n; & (2.3.3) \\ x_{ij} \in \{0,1\} \qquad i = 1, \dots, m, j = 1, \dots, n; & (2.3.4) \end{cases}$$

Compared with the mathematical formulation of GAP, the objective function is different which is usually formulated as a maximization problem. The weight w_{ij} becomes uniform for all the agents and $w_{ij} = 1$. The unit profit p_{ij} is dependent of *i* and *j*. The parameter *i* is allocated to the agent *i*. The parameter *j* is allocated to the group role *j*. The parameters $L[j]$ $(i = 1, \dots, m)$ represent the minimum numbers of agents required for role *j* which are integers.

Since GRAP is a well-known hard problem, it needs advanced methodologies, for example information classification, data mining, pattern search and matching [40].

2.3.3 An Instance of the GRAP

The GRAP has been implemented to a real world problem by Zhu, Zhou and Alkins in 2012 [40]. In a soccer team, there are 20 players ($a_0 \sim a_{19}$) in total. In the field, there are four roles and 11 players for the 1-4-3-3 formation: one goalkeeper (r_0), four backs (r_1), three midfields (r_2), and three forwards (r_3). Figure 11 shows the 20 players and the 4 roles. The coach has to choose 11 players before each game. Players' performance is evaluated by their modes, emotions, health, fatigue, and past performance.

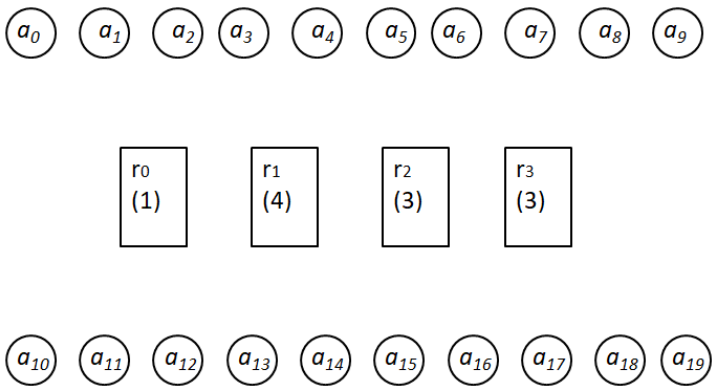


Figure 11: Soccer team [40]

Assume that the coach has the data shown in Figure 12 to represent the players' evaluation values for each role (rows represent players, and columns represent roles). By choosing the players, the coach's goal is to improve the performance of the whole team's by preparing for role assignment.

$$\begin{bmatrix} 0.65 & 0.98 & 0.96 & 0.90 \\ 0.26 & \underline{0.33} & 0.59 & 0.19 \\ 0.72 & 0.61 & 0.19 & 0.63 \\ 0.06 & \underline{0.48} & 0.43 & 0.90 \\ 0.87 & 0.35 & 0.06 & \underline{0.25} \\ \underline{0.72} & 0.15 & 0.28 & 0.01 \\ 0.33 & 0.59 & 0.37 & 0.67 \\ 0.75 & 0.59 & 0.25 & 0.45 \\ 0.12 & 0.10 & 0.01 & 0.51 \\ 0.84 & 0.13 & 0.96 & 0.63 \\ 0.01 & 0.29 & \underline{0.82} & 0.12 \\ 0.07 & 0.52 & 0.36 & 0.95 \\ 0.97 & 0.90 & 0.88 & \underline{0.54} \\ 0.14 & \underline{0.54} & 0.51 & 0.26 \\ 0.04 & 0.03 & 0.83 & 0.70 \\ 0.44 & 0.70 & \underline{0.16} & 0.39 \\ 0.12 & \underline{0.48} & 0.04 & 0.76 \\ 0.30 & 0.14 & 0.52 & 0.08 \\ 0.91 & 0.50 & 0.96 & 0.21 \\ 0.53 & 0.06 & \underline{0.85} & 0.85 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) —— (b)

Figure 12: Evaluation values of agents and roles and the assignment matrix [40]

The team performance is assumed as a simple sum of the selected players' performance on their designated roles. The coach has used several strategies to find the exact optimal solution. From r_0 to r_3, selecting the best players if they have not been selected. Table 1 shows all the group performances based on several strategies.

Strategy	**Assignment for $\{r_0\}\{r_1\}\{r_2\}\{r_3\}$**	**Group Performance**
(r_0, r_1, r_2, r_3)	{12} {0,2,6,15} {9,18,19} {3,11,16}	9.23
(r_0, r_1, r_3, r_2)	{12} {0,2,6,15} {9,14,18} {3,11,19}	9.30
(r_0, r_2, r_1, r_3)	{12} {2,6,7,15} {0,9,18} {3,11,19}	9.04
(r_0, r_2, r_3, r_1)	{12} {2,6,7,15} {0,9,18} {3,11,19}	9.04
(r_0, r_3, r_1, r_2)	{12} {2,6,7,15} {9,18,19} {0,3,11}	8.98
(r_0, r_3, r_2, r_1)	{12} {2,6,7,15} {9,18,19} {0,3,11}	8.98
(r_1, r_0, r_2, r_3)	{18} {0,2,12,15} {9,14,19} {0,3,11}	9.35
(r_1, r_0, r_3, r_2)	{18} {0,2,12,15} {9,10,14} {3,11,16}	9.41
(r_1, r_2, r_0, r_3)	{4} {0,2,12,15} {9,18,19} {3,11,16}	9.44
(r_1, r_2, r_3, r_0)	{4} {0,2,12,15} {9,18,19} {3,11,16}	9.44
(r_1, r_3, r_0, r_2)	{18} {0,2,12,15} {9,10,14} {3,11,19}	9.41
$(\boldsymbol{r_1, r_3, r_2, r_0})$	$\{\mathbf{4}\}\{\mathbf{0,2,12,15}\}\{\mathbf{9,14,18}\}\{\mathbf{3,11,19}\}$	**9.51**
(r_2, r_0, r_1, r_3)	{12} {2,6,7,15} {0,9,18} {3,11,19}	9.04
(r_2, r_0, r_3, r_1)	{12} {2,6,7,15} {0,9,18} {3,11,19}	9.04
(r_2, r_1, r_0, r_3)	{4} {2,6,12,15} {0,9,18} {3,11,19}	9.25

(r_2, r_1, r_3, r_0)	{4} {2,6,12,15} {0,9,18} {3,11,19}	9.25
(r_2, r_3, r_0, r_2)	{12} {0,9,18} {1,10,14} {3,11,19}	8.79
(r_2, r_3, r_2, r_0)	{4} {0,9,18} {10,12,14} {3,11,19}	8.98
(r_3, r_0, r_1, r_2)	{12} {2,6,7,15} {9,18,19} {0,3,11}	8.98
(r_3, r_0, r_2, r_1)	{12} {2,6,7,15} {9,18,19} {0,3,11}	8.98
(r_3, r_1, r_0, r_2)	{18} {2,6,12,15} {9,14,19} {0,3,11}	9.10
(r_3, r_1, r_2, r_0)	{4} {2,6,12,15} {9,18,19} {0,3,11}	9.19
(r_3, r_2, r_0, r_1)	{4} {2,6,7,15} {9,12,18} {0,3,11}	8.91
(r_3, r_2, r_1, r_0)	{4} {2,6,7,15} {9,12,18} {0,3,11}	8.91

Table 1:

Comparisons among assignment strategies [40]

By using the enumeration method, the optimal solution is (r_1, r_3, r_2, r_0) as shown in bold row in Table 1. The solution is shown in Figure 12(a) as circles, in Figure 12(b) as a matrix, and in Figure 13 as a graph.

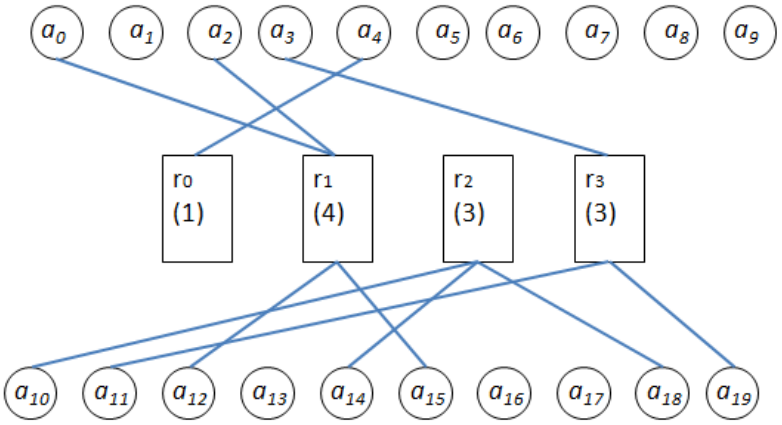


Figure 13: Optimal solution [40]

2.3.4 Concepts

In formalizing GRAPs, m expresses the size of the agent, and n expresses the size of the role. For example, in the soccer team, m is 20 players, n is four roles as goalkeeper, backs, midfields, and forwards.

Role Range Vector: A role range vector is a vector of the lower ranges of roles. The role range vector is denoted as $L[j] \in N$, where N is the set of natural numbers and $0 \leq j < n$. For example, $L = [1, 4, 3, 3]$ for the soccer team in Fig. 11.

Qualification Matrix: The qualification matrix is an $m \times n$ matrix of values in $[0, 1]$,where $Q[i, j]$ expresses the qualification value of agent i for role j. It is denoted as $Q[i, j] \in [0, 1](0 \leq i < m; 0 \leq j < n)$. For example, Fig.12(a) shows a qualification matrix for the soccer team.

Role Assignment Matrix: A role assignment matrix is an $m \times n$ matrix of values in $\{0, 1\}$. If $T[i, j] = 1$, agent i is assigned to role j, and agent i is called an assigned agent. It is denoted as $T[i, j] \in \{0, 1\}(0 \leq i < m; 0 \leq j < n)$. For example, Fig.12(b) shows an assignment matrix for the soccer team.

Group Qualification: A group qualification is defined as the sum of the assigned agents' qualifications, i.e. $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j]$. For example, the group qualification of Fig.12 is 9.51.

Role Weight Vector: A role weight vector is a vector of the weights of roles. The role weight vector is denoted as $W[j] \in [0, 1](0 \leq j < n)$. For example, for the soccer team in Fig. 2, if the attack is emphasized, the set $W = [0.6, 0.7, 0.8, 0.9]$, but if the defense is emphasized, the set $W = [0.9, 0.8, 0.7, 0.6]$.

Weighted Group Qualification: A weighted group qualification is defined as the weighted sum of assigned agents' qualifications, i.e., $\sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i, j] \times T[i, j]$. For example, for the assignment in Fig. 12, if $W = [0.6, 0.7, 0.8, 0.9]$, the weighted group qualification is 7.2, and the assignment shown in Fig. 12(b) is not optimal in this situation.

2.3.5 Solution to GRAP

In order to apply the K-M algorithm into solving the GRAPs, many aspects should be considered. Given a matrix which has i agents and j roles, where 0 and 1 mean no and yes, respectively. The roles are equally important. Based on different requirements, GRAP can be categorized into different levels:

- Simple Role Assignment Problem (SGRAP): to find a role assignment matrix T that makes it workable;
- Rated Role Assignment Problem (RGRAP): favors a solution with the maximum overall sum of qualifications for all assigned agents;
- Weighted Role Assignment Problem (WGRAP): if the roles in a group have different importance, for example, the roles have different weights.

Condition 1: The total agent number should be larger than the required numbers, $m > \sum_{j=0}^{n-1} L[j]$.

Condition 2: For each role, there should be enough qualified agents, $\sum_{i=0}^{m-1}[Q[i,j] - \tau] \geq L[j]$, where $0 \leq j < n$.

Qualification Threshold (τ): τ is a real number in [0, 1] to state that an agent is called qualified in a group only if its qualification value is greater than τ. If an agent's qualification value is not greater than τ, the agent is unqualified for the role.

Role Index Vector L': A role index vector is an $m(> \sum_{j=0}^{n-1} L[j])$ dimensional vector created from a role range vector L. $L'[k]$ is a role number related with column $k(0 \leq k < m)$ in the adjusted qualification matrix Q' defined below, where

$$L'[k] = \begin{cases} 0 & (k < L[0]) \\ x & (\sum_{p=0}^{x-1} L[p] \leq k < \sum_{p=0}^{x} L[p](0 < x < n)) \\ n & (k \geq \sum_{p=0}^{n-1} L[p]) \end{cases}$$

Adjusted Qualification Matrix Q': Q' is an $m \times m$ matrix, where $Q'[i,j] \in [0,1]$ expresses the qualification value of agent i for role $L'[j]$, where

$$Q'[i,j] = \begin{cases} Q'[i, L'[j]] & (0 \leq i < m; 0 \leq j < \sum_{p=0}^{n-1} L[p]) \\ 1 & (0 \leq i < m; \sum_{p=0}^{n-1} L[p] \leq j < m) \end{cases}$$

By adjusting the number of agents and roles, the RGRAP can be transferred to a GAP. GAP has been solved by K-M algorithm. Since K-M algorithm solve the square matrix of GAP, which means the number of agents should be equal to that of roles. In the real world scenario, the assignments will not always meet this condition. Obviously, we need to transfer the input matrix into a square matrix. In order to form the square matrix, they duplicate rows or columns by using the Role Range Vector, Qualification Threshold, Role Index Vector and other definitions and conditions to transfer RGRAP to GAP, as well as avoid the incorrect optimal result. For example, for $Q(L = [1, 2, 1])$ in Figure 14, its square matrix is shown in Figure 15. If $\tau = 0.6$, its adjusting matrix Q' is shown in Figure 16 and the relevant coumn number vector $L' = [0, 1, 1, 2, 3, 3]$, where 3 corresponds to empty roles.

$$\begin{bmatrix} 0.36 & 0.76 & 0.72 \\ 0.93 & 0.59 & 0.24 \\ 0.06 & 0.46 & 0.69 \\ 0.40 & 0.10 & 0.74 \\ 0.23 & 0.75 & 0.24 \\ 0.21 & 0.77 & 0.24 \end{bmatrix}$$

Figure 14: Matrix with

$$\boldsymbol{m > \sum_{j=0}^{n-1} L[j]}$$

$$\begin{bmatrix} 0.36 & 0.76 & 0.76 & 0.72 & 1.0 & 1.0 \\ 0.93 & 0.59 & 0.59 & 0.24 & 1.0 & 1.0 \\ 0.06 & 0.46 & 0.46 & 0.69 & 1.0 & 1.0 \\ 0.40 & 0.10 & 0.10 & 0.74 & 1.0 & 1.0 \\ 0.23 & 0.75 & 0.75 & 0.24 & 1.0 & 1.0 \\ 0.21 & 0.77 & 0.77 & 0.24 & 1.0 & 1.0 \end{bmatrix}$$

Figure 15: Created square matrix

$$\begin{bmatrix} -36 & 0.76 & 0.76 & 0.72 & 1.0 & 1.0 \\ 0.93 & -36 & -36 & -36 & 1.0 & 1.0 \\ -36 & -36 & -36 & 0.69 & 1.0 & 1.0 \\ -36 & -36 & -36 & 0.74 & 1.0 & 1.0 \\ -36 & 0.75 & 0.75 & -36 & 1.0 & 1.0 \\ -36 & 0.77 & 0.77 & -36 & 1.0 & 1.0 \end{bmatrix}$$

Figure 16: Adjusting square matrix

Because the K-M algorithm solves the minimization problem, it is easy to transfer it to a maximization one. Subtracting the entries of Q' from the largest entry of Q' to obtain a new matrix M. For example, for the maximization of Q' in Figure 16, a matrix M (Figure 17) can be obtained by using the equation: $M[i,j] = 1 - Q'[i,j], (0 \leq i,j < m)$. After calling the K-M algorithm, the final assignment is produced based on the result (Figure 18). The group qualification is 3.2.

$$\begin{bmatrix} 37 & 0.24 & 0.24 & 0.28 & 0.0 & 0.0 \\ 0..07 & 37 & 37 & 37 & 0.0 & 0.0 \\ 37 & 37 & 37 & 0.31 & 0.0 & 0.0 \\ 37 & 37 & 37 & 0.26 & 0.0 & 0.0 \\ 37 & 0.25 & 0.25 & 37 & 0.0 & 0.0 \\ 37 & 0.23 & 0.23 & 37 & 0.0 & 0.0 \end{bmatrix}$$

Figure 17: Square matrix transferred from the qualification matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 18: Assignment Matrix $\boldsymbol{T}$

In the final part, they presented the numerical results of different sizes of assignments. The group size varied from 10 to 100, with the efficiency of K-M algorithm, it can solve the assignment within 40 milliseconds on the provided simulation platform. The computational complexity of the K-M algorithm is $O(n^3)$.

Chapter 3

3 Incremental Assignment Problem

This chapter is a review of the Incremental Assignment Problem (IAP). It includes:

- The introduction of Incremental Assignment Problem (IAP),
- The related work of IAP, and
- The IAP Algorithm.

3.1 Introduction

The assignment problem is a basic combinatorial optimization problem. It involves finding a matching in a weighted bipartite graph where the sum of weights of the edges is as large as possible. It is a widely-studied problem that applied to many areas [85]. A common variant includes finding a minimum-weight perfect matching. It can be expressed as follows: given a group of workers, a set of jobs, and a set of ratings indicating how well each worker can perform each job, determine the best work assignment to maximize the overall rating [80]. More generally, given a bipartite graph consisting of two partitions U and V, and a set of weighted edges E between the two partitions, the problem requires selecting a subset of the edges with a maximum sum of weights such that each node $u_i \in U, v_j \in V$ is connected to at most one edge. By considering a set of non-negative edge costs, $c_{ij} = W - w_{ij}$, the problem can also be stated as a minimization problem, instead of edge weights w_{ij}, where W is at least as large as the maximum of all the edge weights.

A matching or independent edge set in a graph is a set of edges that have no common vertices. There are several algorithms of the matching problem, such as matching in weighted bipartite graphs, matching in unweighted graphs, matching in general graphs. In unweighted graphs maximum cardinality matching is sought [77]. As for matching of weighted bipartite graphs, each edge has a correlation value. A maximum weighted bipartite matching is defined as a matching where the sum of the values of the edges in the matching has a maximal value. If the graph is incomplete, the missing edge is inserted with value zero. Finding such a matching is called the assignment problem. A common variant involves finding a minimum-weighted perfect matching [78]. Kuhn-Munkres algorithm or the Hungarian algorithm is the well-known algorithm for the assignment problem, originally proposed by H.W.Kuhn in 1955 [79] and refined by J. Munkres in 1957 [80]. When it is implemented with proper data structures, this algorithm has $O(n^3)$ complexity.

The incremental assignment problem is described as: given a weighted bipartite graph and its maximum weighted matching, determine the maximum weighted matching of the graph extended with a new pair of vertices, one per partition, and the weighted edges connecting these new vertices to all the vertices on their opposite partitions [81]. The Hungarian Algorithm can

solve it as the ordinary assignment problem. But Toroslu and Üçoluk, propose an algorithm that uses the given maximum weighted matching of the maximum-weighted-matched part of the bipartite graph in order to determine the maximum weighted matching of the entire (extended) bipartite graph. The complexity of the algorithm is $O(n^2)$.

This chapter is organized as follows. It first discusses the related work. Then we will present the algorithm of the incremental assignment problem and the example.

3.2 Related Work

3.2.1 An addendum and the authors' response to the addendum

An addendum on the incremental assignment problem by A.Volgenant [82] points out that the algorithm of the incremental assignment problem can be found in the literature. Tomizawa proposes an algorithm to solve transportation problems and assignment problem [27]. It solves a successive series of incremented sub-problems and gives an illustrative example. Jonker and Bolgenant improved the Hungarian algorithm that labels rows one by one [20]. This idea is "equivalent to solve an $k \times k$ assignment problem ($2 \leq k \leq n$), when a partial solution of $k-1$ assignment is available". The decremental assignment problem is also presented by Toroslu and Üçoluk. Volgenant pointed out that it is equivalent to appropriately changing the weight coefficients of the arcs incident to the pair of vertices and re-optimize the problem in $O(n^2)$ which takes two steps of Ahuja's [84] shortest augmenting path method.

Toroslu and Üçoluk respond to Volgenant [83]. They state that the two papers [8,9] use some incremental methods to solve the standard assignment problem, Toroslu and Üçoluk, on the other hand, "introduce this new problem, proposes an algorithm specific to this problem and studies the complexity and the correctness of the solution in detail" [83]. The papers do not formally address the complexity and the correctness of incremental steps of their algorithm. The paper [27] focuses on different improvements using the shortest augmenting paths and studies the performance of these improvements empirically. Jonker and Bolgenant [20] present a completely different problem, the transportation network problem. They informally define the assignment problem and then describe an incremental solution. This is just a sketch of the idea

and illustrates an example, but they do not describe the detailed algorithm or the proof of complexity and correctness.

3.2.2 The Dynamic Hungarian Algorithm

Mills-Tettey, Stentz and Dias [19] present the dynamic Hungarian algorithm to solving the assignment problem by changing edge costs or weight. They propose the new version of dynamic algorithm that is more efficient by fixing the initial solution obtained before the cost changes.

They extend the idea in incremental assignment problem to handle multiple updates and address deletions (assuming one-to-one assignment). For example, consider a problem where the nodes in the graph represent workers and jobs to be performed by these workers, and the edges represent transportation costs between the worker locations and job locations. In this area, a given road may be closed unexpectedly, greatly increasing the costs of some workers completing certain jobs.

They present proofs of the correctness and efficiency of their algorithm and provide simulation results that illustrate its efficiency. The complexity of the algorithm is $O(kn^2)$ where n is the size of one partition of the bipartite graph, and k is the number of changed rows or columns in the cost matrix.

3.3 The Algorithm for Incremental Assignment Problem

Toroslu and Üçoluk examine the incremental assignment problem and present an elegant algorithm for its solution. In the incremental assignment problem, a weighted bipartite graph and its maximum weighted matching are given, we want to determine the maximum weighted matching of the graph extended with a new pair of vertices (an adding row and an adding column), one on each partition, and weighted edges connecting these new vertices to all the vertices on their opposite partitions [81].

3.3.1 The Algorithm

The algorithm assigns dual variables α_i to each node U and dual variables β_j to each node V. The assignment problem is feasible when $\alpha_i + \beta_j \geq W_{ij}$. The Hungarian algorithm

maintains feasible values for all the α_i and β_j from initialization through termination. An edge in the bipartite graph is called admissible when $\alpha_i + \beta_j = W_{ij}$.

The algorithm first determines the feasible values for the two new dual variables α_{n+1} and β_{n+1} (the other dual variables are still feasible from the solution of the $n \times n$ problem). It then essentially performs a single stage of the Hungarian algorithm to find an augmenting path between the two new vertices U_{n+1} and V_{n+1}, adjusting dual variables to add new admissible edges to the equality subgraph as needed. The matched and unmatched edges are flipped along the discovered augmenting path increases the cardinality of the matching by one, thus resulting in a complete matching. Note that the two new nodes u_{n+1} and v_{n+1} may not need to be matched to each other in the extended matching. Because the algorithm involves executing only one stage of the Hungarian algorithm after performing an $O(n)$ initialization step, the computational complexity of the incremental assignment algorithm is $O(n^2)$. The incremental assignment algorithm is illustrated as follows.

Incremental Assignment Algorithm:

Input:

- An assignment problem comprising a bipartite graph, $\{U, V; E\}$ (where $|U| = |V| = n + 1$) and an $(n + 1) \times (n + 1)$ matrix of edge weights W_{ij}
- An optimal solution to the $n \times n$ sub-problem of the above assignment problem, comprising a matching M^* of the first n nodes of U to the first n nodes of V, and the final values of the dual variables α_i and β_j for $i \in 1 \ldots n$ and $j \in i \ldots n$

Output: An optimal matching M, for the $(n + 1) \times (n + 1)$ problem.

---

1. Perform initialization:
   (a) Begin with the given matching, $M_0 = M^*$.
   (b) Assign feasible values to the dual variables α_{n+1} and β_{n+1} as follows:

$$\beta_{n+1} = max \ \ (max_{1 \le i \le n}(W_{i(n+1)} - \alpha_i), W_{(n+1)(n+1)}) \qquad (3.3.1.1)$$

$$\alpha_{n+1} = max_{1 \le j \le n+1}(W_{(n+1)j} - \beta_j) \qquad (3.3.1.2)$$

2. Perform the routine **Stage** below.
3. Output the resulting matching M.

Stage:

1. Designate each exposed (unmatched) node in U as the root of a Hungarian tree.

2. Grow the Hungarian tree rooted at the exposed nodes in the equality subgraph. Designate the indices i of nodes u_i encountered in the Hungarian tree by the set I^*, and the indices j of nodes v_j encountered in the Hungarian tree by the set J^*. If an augmenting path is found, go to **Step 4**. If not, and the Hungarian trees cannot be grown further, proceed to **Step 3**.

3. Modify the dual variables α_i and β_j as follows to add new edges to the equality subgraph. Then go to **Step 2** to continue the search for the augmenting path.

$$\theta = \min_{i \in I^*, j \notin J^*} (\alpha_i + \beta_j - W_{ij})$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i - \theta & i \in I^* \\ \alpha_i & i \notin I^* \end{cases}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J^* \\ \beta_j & j \notin J^* \end{cases}$$

4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path. That is, M_k (the new matching at stage k) is given by $(M_{k-1} - P) \cup (P - M_{k-1})$, where M_{k-1} is the matching from the previous stage and P is the set of edges on the selected augmenting path.

3.3.2 Example

We propose the same example which presented by Toroslu and Üçoluk. Consider the 4×4 weighted bipartite graph described by its weight matrix as follows:

					α_i
	V_1	V_2	V_3	V_4	$\downarrow$
U_1	5	1	1	1	0
U_2	4	3	1	3	-1
U_3	5	4	3	4	0
U_4	1	6	2	5	2
$\beta_j \rightarrow$	5	4	3	5	

Figure 19: **4** × **4** matrix

Assume that the first 3 vertices are given, which corresponds to the diagonal elements of the weight matrix. The last row and column correspond to the newly added vertex pairs. First two new vertices are added (U_4 and V_4). The dual variables α_4 and β_4 are calculated by using the equation (3.3.1.1) and (3.3.1.2) as shown above. The Hungarian tree is obtained and no augmenting path is found (Figure 21).

					α_i
	V_1	V_2	V_3	V_4	↓
U_1	5	1	1	1	0
U_2	4	3	1	3	-1
U_3	5	4	3	4	0
U_4	1	6	2	5	2
β_j	→ 5	4	3	5	

Figure 20: Situation before the first iteration of the algorithm: Weight Matrix

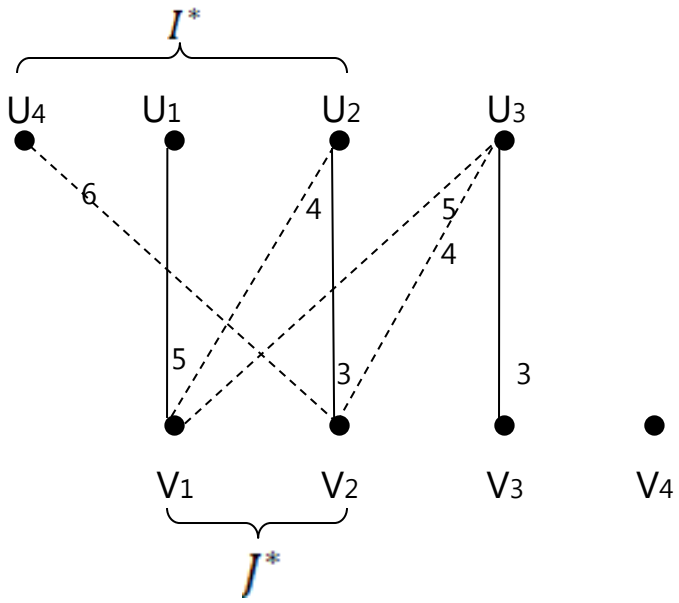


Figure 21: Situation before the first iteration of the algorithm: Equality Subgraph

Revise the dual variables by using Stage→Step 3 in order to preserve all the matching edges, while adding new edges to M_k. By calculating, $\theta = 1$, then reducing the dual variables from the nodes I^* by 1, increasing the dual variables from the nodes J^* by 1 (Figure 22). Then

adjusting dual variables to add new admissible edges to the equality subgraph as needed (Figure 23).

		V_1	V_2	V_3	V_4	α_i ↓		α_i
	U_1	5	1	1	1	0	(-1)	-1
	U_2	4	3	1	3	-1	(-1)	-2
	U_3	5	4	3	4	0		0
	U_4	1	6	2	5	2	(-1)	1
β_j	→	5	4	3	5			
		(+1)	(+1)					
β_j		6	5	3	5			

Figure 22: Situation after the first iteration of the algorithm: Weight Matrix

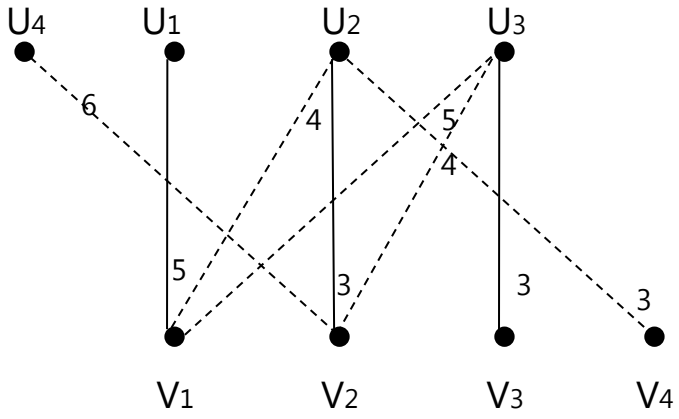


Figure 23: Situation after the first iteration of the algorithm: Equality Subgraph

The augmenting path has been found as U_4, V_2, U_3. The matched and unmatched edges flip along the discovered augmenting path increases the cardinality of the matching by one, thus resulting in a complete matching. This matching is perfect, and hence it must be the optimal. The matching (U_1, V_1), (U_2, V_4), (U_3, V_3), (U_4, V_2) has cost 5+6+3+3=17 which is exactly the sum of the labels in the dual variables.

Chapter 4

4 Improved Incremental Assignment Algorithm

This chapter will provide an improved algorithm for Incremental Assignment Problem.

This chapter will present:

- The requirement of Improved Incremental Assignment Algorithm,
- The Improved Incremental Assignment Algorithm,
- The overall program flow chart,
- The platform of simulation, and
- Simulation results and analysis.

4.1 Introduction

The incremental assignment problem is described as: given a weighted bipartite graph and its maximum weighted matching, determine the maximum weighted matching of the graph extended by a new pair of vertices, one on each partition, and weighted edges connecting these new vertices to all the vertices on their opposite partitions [81]. It can be solved with the Hungarian Algorithm as the general assignment problem. But in Toroslu's work, they propose an algorithm that uses the given maximum weighted matching of the maximum-weighted-matched part of the bipartite graph to determine the maximum weighted matching of the entire (extended) bipartite graph. The complexity of the algorithm is $O(n^2)$.

Considering there will be thousands or millions weights. It is costly to calculate by using incremental assignment algorithm. The goal is to propose an algorithm to improve the incremental assignment algorithm.

4.2 Improved Incremental Assignment Algorithm

The algorithm first determines the feasible values of the two new dual variables α_{n+1} and β_{n+1} (the other dual variables are still feasible from the solution of the $n \times n$ matrix). It then basically performs a single stage of the Hungarian algorithm to find an augmenting path between the two new vertices U_{n+1} and V_{n+1}.

DC_k is the maximum value of the difference between the column weight and its dual variables;

DR_t is the maximum value of the difference between the row weight and its dual variables;

$X_{kt} = 1$ means W_{kt} is the optimal solution of the kth row tth column;

$X_{kt} \neq 1$ means W_{kt} is not the optimal solution of the kth row tth column.

- If $X_{kt} = 1$, calculating DC_k and DR_t by using the dual variables α_n and β_n, then compare the sum (I) of DC_k and DR_t with the weight $W_{(n+1)(n+1)}$. Then get the optimal matching.
- If $X_{kt} \neq 1$, adjusting dual variables to add new admissible edges to the equality subgraph as

needed. Flipping the matched and unmatched edges along the discovered augmenting path increases the cardinality of the matching by one, thus resulting in a complete matching. Note that the two new nodes u_{n+1} and v_{n+1} may not necessarily be matched to each other in the final matching. The improved incremental assignment algorithm is illustrated below. Figure 24 shows the overall program flow chart.

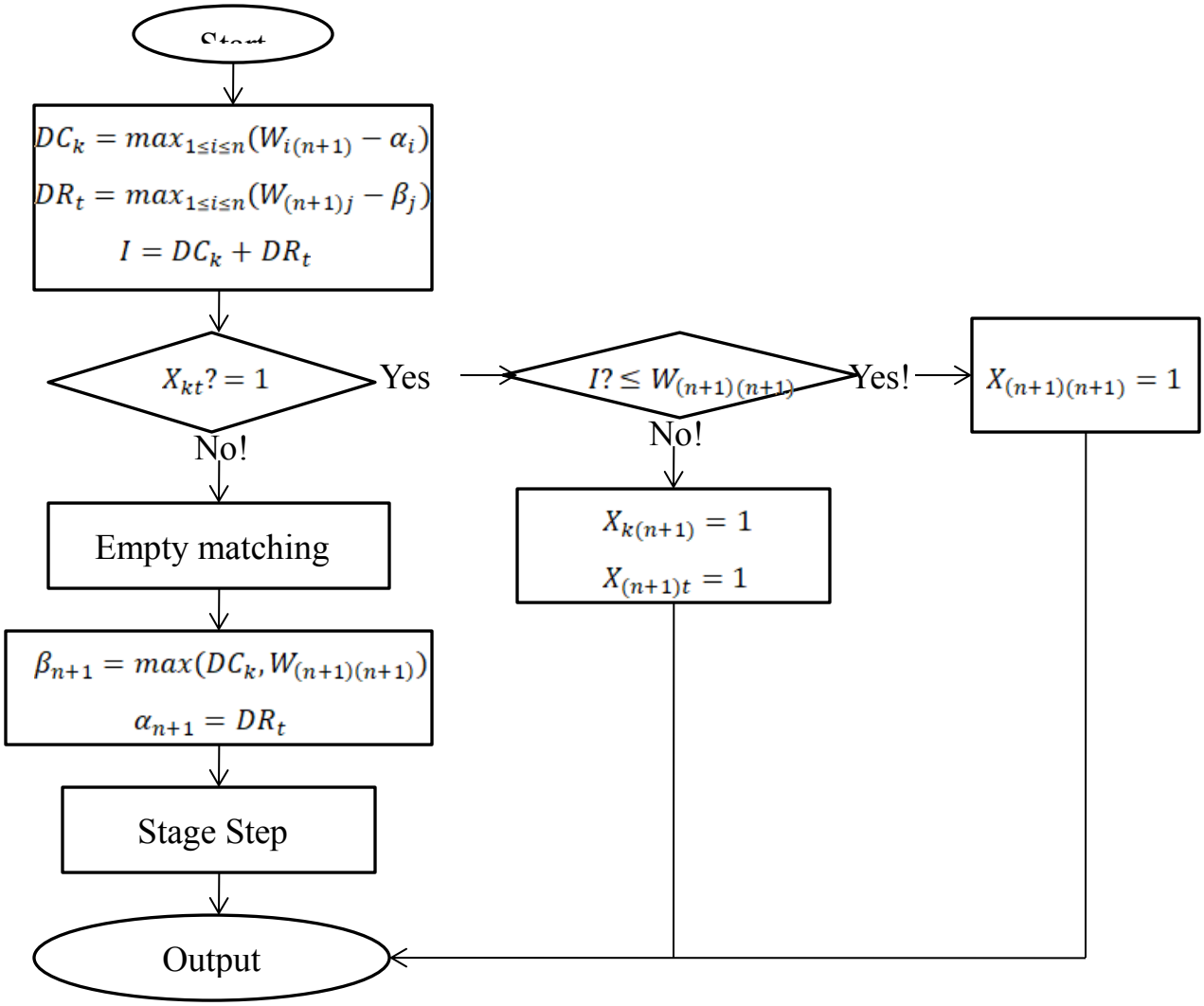


Figure 24: The overall program flow chart

Improved Incremental Assignment Algorithm:

Input:

- An assignment problem comprising a bipartite graph, $\{V, E\}$ (where $V = X \cup Y, X \cap Y = \emptyset,\ |X| = |Y| = n + 1$) and an $(n + 1) \times (n + 1)$ matrix of edge weights W_{ij}
- An optimal solution to the $n \times n$ sub-problem of the above assignment problem, comprising a matching M^* of the first n nodes of X to the first n nodes of Y, and the final values of the dual variables α_i and β_j for $i \in 1 \ldots n$ and $j \in i \ldots n$

Output: An optimal matching M, for the $(n + 1) \times (n + 1)$ problem.

1. Perform initialization:
   (a) Find the difference of each role and each column as follows:

$$DC_k = max_{1 \le i \le n}(W_{i(n+1)} - \alpha_i) \quad (4.2.1)$$

$$DR_t = max_{1 \le i \le n}(W_{(n+1)j} - \beta_j) \quad (4.2.2)$$

$$I = DC_k + DR_t \quad (4.2.3)$$

   (b) If $X_{kt} = 1$,
      i. $I \le W_{(n+1)(n+1)}$, then
         $X_{(n+1)(n+1)} = 1$, then
         go to step 3.
      ii. $I > W_{(n+1)(n+1)}$, then
         find $W_{k(n+1)}$ and $W_{(n+1)t}$ such $X_{k(n+1)} = 1$ and $X_{(n+1)t} = 1$, then
         go to step 3.
   (c) Else if $X_{kt} \neq 1$, Begin with an empty matching, $M_0 = M^*$. Assign feasible values to the dual variables α_{n+1} and β_{n+1} as follows:

$$\beta_{n+1} = max\ \ (DC_k, W_{(n+1)(n+1)}) \quad (4.2.4)$$

$$\alpha_{n+1} = DR_t \quad (4.2.5)$$

2. Perform the **Stage** from the basic Hungarian algorithm detailed in the Hungarian Algorithm.
3. Output the resulting matching M.

Stage:

1. Designate each exposed (unmatched) node in X as the root of a Hungarian tree.

2. Grow the Hungarian tree rooted at the exposed nodes in the equality subgraph. Designate the indices i of nodes x_i encountered in the Hungarian tree by the set I^*, and the indices j of nodes y_j encountered in the Hungarian tee by the set J^*. If an augmenting path is found, go to step (4). If not, and the Hungarian trees cannot be grown further, proceed to step (3).

3. Modify the dual variables α_i and β_j as follows to add new edges to the equality subgraph. Then go to step (2) to continue the search for the augmenting path.

$$\theta = \min_{i \in I^*, j \notin J^*} (\alpha_i + \beta_j - W_{ij})$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i - \theta & i \in I^* \\ \alpha_i & i \notin I^* \end{cases}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J^* \\ \beta_j & j \notin J^* \end{cases}$$

4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path. That is, M_k (the new matching at stage k) is given by $(M_{k-1} - P) \cup (P - M_{k-1})$, where M_{k-1} is the matching from the previous stage and P is the set of edges on the selected augmenting path.

4.3 Examples

Consider that we are given the assignment among the first 3 vertices, which corresponds to the diagonal elements of the weight matrix. The optimal matching and the value of dual variables are given.

	Y_1	Y_2	Y_3	α_i
X_1	(5)	1	1	0
X_2	4	(3)	1	-1
X_3	5	4	(3)	0

β_j	5	4	3

Figure 25: $\mathbf{3 \times 3}$ matrix

The examples below show that how the algorithm will be implemented under different cases. The last row and column are the newly added vertex pairs.

Example 1: Consider the 4×4 weighted bipartite graph described by its weight matrix as follows:

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	5	1	1	1	0
X_2	4	3	1	3	-1
X_3	5	4	3	3	0
X_4	1	5	2	6	
β_j	5	4	3		

Figure 26: 4×4 matrix

By using the Equation (4.2.1), (4.2.2) and (4.2.3), the largest difference of the column is $DC_2 = 4$, the largest difference of the row is $DR_2 = 1$, then $I = 4 + 1 = 5$. As $W_{44} = 6$, then $I < W_{44}$. As $X_{22} = 1$, then $X_{44} = 1$. The optimal solution is 5+3+3+6=17.

Example 2: Consider the 4×4 weighted bipartite graph described by its weight matrix as follows:

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	5	1	1	1	0
X_2	4	3	1	3	-1
X_3	5	4	3	3	0
X_4	1	6	2	5	
β_j	5	4	3		

Figure 27: 4×4 matrix

This is the same example shown in [81]. We do not need to apply the Hungarian algorithm to find the dual variables. As the Improved Incremental Assignment Algorithm above, by using the Equation (4.2.1), (4.2.2) and (4.2.3), the largest difference of the column is $DC_2 = 4$, the largest difference of the row is $DR_2 = 2$, then $I = 4 + 2 = 6$. As $W_{44} = 5$, then $I > W_{44}$. As $X_{22} = 1$, then $X_{24} = 1$ and $X_{42} = 1$. The optimal solution is 5+3+3+6=17.

Example 3: Consider the 4 × 4 weighted bipartite graph described by its weight matrix as follows:

					α_i
	Y_1	Y_2	Y_3	Y_4	
X_1	5	1	1	1	0
X_2	4	3	1	3	-1
X_3	5	4	3	3	0
X_4	1	3	6	5	
β_j	5	4	3		

Figure 28: **4** × **4** matrix

As the Improved Incremental Assignment Algorithm above, by using the Equation (4.2.1), (4.2.2) and (4.2.3), the largest difference of the column is $DC_2 = 4$, the largest difference of the row is $DR_3 = 3$. But $X_{23} \neq 1$, Perform the **Stage** from the basic Hungarian algorithm. Determine the augmenting path on the weighted bipartite graph which was extended by the feasible vertex labels.

					α_i
	Y_1	Y_2	Y_3	Y_4	
X_1	**5**	1	1	1	0
X_2	4	**3**	1	3	-1
X_3	5	4	**3**	3	0
X_4	1	3	6	5	3

β_j	5	4	3	5

Figure 29: Weight Matrix

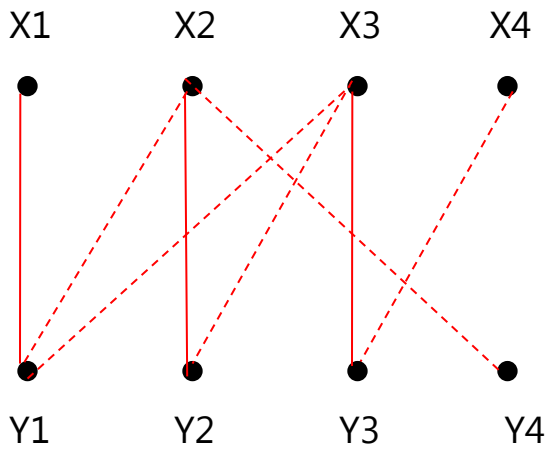


Figure 30: Equality subgraph

The weighted matrix presents the feasible vertex labels of its vertices. The equality subgraph obtained from the weight matrix and the feasible vertex label is shown in Figure 30.

Augmenting path $X_4, Y_3, X_3, Y_2, X_2, Y_4$ is found; interchange matched and unmatched edges in the augmenting path. The optimal solution (W_{11}, W_{32}, W_{43} and W_{24}) has been found which is 5+3+4+6=18.

4.4 Platform of Simulation

The hardware platform is a laptop with a CPU of 2.30GHz and a main memory of 8GB. The development environment is Microsoft Windows 10 (Home Edition) and Eclipse 4.11.

The simulation result is presented in 4.5.

4.5 Implementation and Performance Experiments

When properly implemented, the Incremental Assignment Algorithm (IAA) can operate with the computational complexity of $O(n^2)$ [81]. To verify the performance of the Improved Incremental Assignment Algorithm (IIAA), a program is implemented based on that of the IAA.

Three cases are designed for the Improved Incremental Assignment Algorithm (IIAA) in different dimensions, where the percentage tells the operation time of Improved Incremental Assignment Algorithm time as a percentage of Incremental Assignment Algorithm among 600 random matrixes. Case 1 shows the operation time comparison of the circumstances that $X_{kt} = 1$ and $I > W_{(n+1)(n+1)}$. Case 2 shows the running time comparison of the matrix that $X_{kt} = 1$ and $I \leq W_{(n+1)(n+1)}$. Case 3 is the general cases which will express the chances that a random matrix will be solved by the IIAA and how much percentage of the operation time will be saved.

All the experiments are workable in both algorithms. Each matrix is formed by randomly creating weights. Each case takes 20 different dimensions which the largest is 100 dimensions (i.e., a matrix with 100×100 elements) and the smallest is 5 dimensions (a matrix with 5×5 elements). Each dimension repeats 10 times to show that the algorithm is workable.

Table 2

The Operation Time for Case 1

Time (ms) / Dimension	Improved IAA Average	IAA Average	Percentage
5	1.648110	7.336810	22%
10	1.828240	12.697650	14%
15	1.694130	8.413910	20%
20	1.809400	12.952850	14%
25	1.222370	13.128120	9%
30	2.203930	35.669240	6%
35	1.765967	20.204889	9%
40	1.904920	42.730290	4%
45	2.095520	34.807790	6%
50	1.731450	48.148740	4%
55	1.928090	52.943150	4%
60	2.308790	73.964380	3%
65	3.248080	76.252410	4%
70	3.240990	99.721670	3%
75	3.350340	81.735200	4%
80	4.353430	87.974030	5%

85	2.976130	91.464860	3%
90	3.102580	98.607320	3%
95	3.155540	101.195130	3%
100	4.197170	106.825880	4%

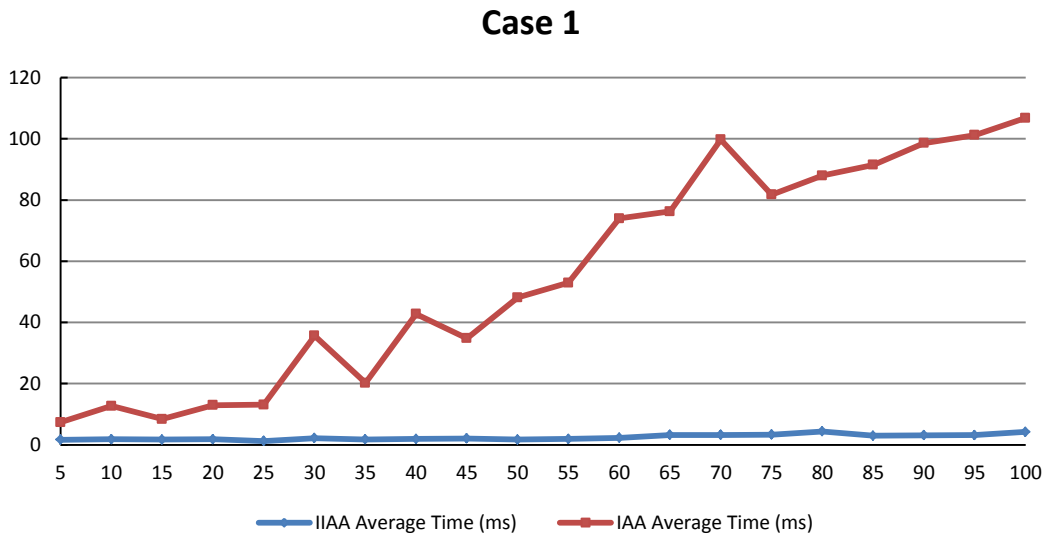


Figure 31: Trend lines for average operation time for different dimensions

Table 1 presents the average time of different dimension matrix which has been solved by Improved Incremental Assignment Algorithm and Incremental Assignment Algorithm for Case 1. Timing unit is millisecond.

$$\text{Percentage} = \frac{\text{Improved IAA average time}}{\text{IAA average time}}$$

Figure 31 shows the trend line for average operation time for different dimensions in Case 1.

Table 3

The Operation Time for Case 2

Time (ms) / Dimension	Improved IAA Average	IAA Average	Percentage
5	1.529310	5.305950	29%
10	1.833020	14.519800	13%
15	1.533510	10.288430	15%
20	1.212810	15.268100	8%
25	1.110590	19.689600	6%
30	1.772970	29.741750	6%
35	1.399510	24.993040	6%
40	2.009330	46.639780	4%
45	1.584130	36.025330	4%
50	2.325630	67.057310	3%
55	1.879350	54.300640	3%
60	2.291460	78.948510	3%
65	3.270390	92.805720	4%
70	3.208170	100.959030	3%
75	4.811270	83.651650	6%
80	4.267130	91.528590	5%
85	3.237480	100.349570	3%
90	2.864940	99.996790	3%
95	3.298380	105.278890	3%
100	3.993250	118.730300	3%

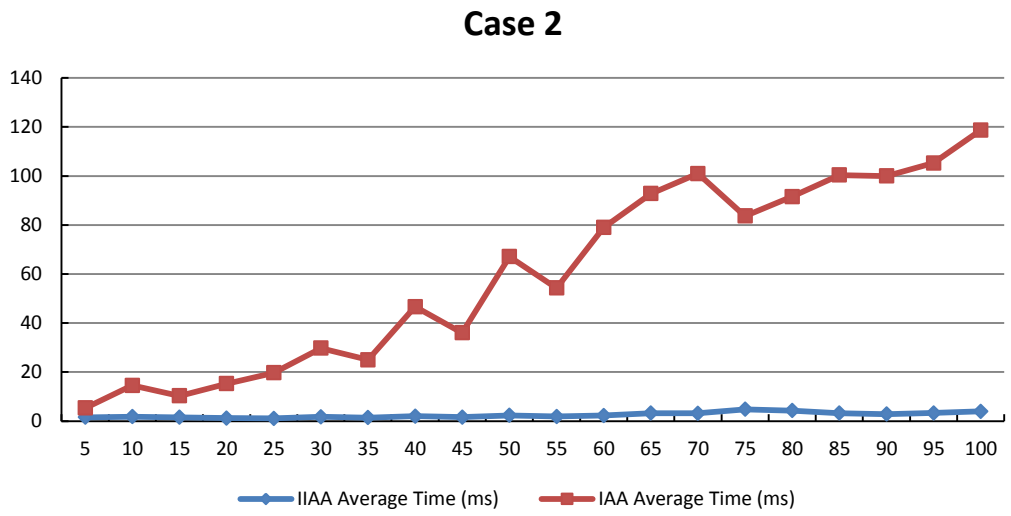


Figure 32: Trend lines for operation time for different dimensions

Table 3 presents the average time of different dimension matrix which has been solved by Improved Incremental Assignment Algorithm and Incremental Assignment Algorithm for Case 2. Timing unit is millisecond.

Figure 32 shows the trend line for average operation time for different dimensions in Case 2.

Table 4

The Operation Time for Case 3 (General Case)

Time / Dimension	Average IIAA	Average IAA	Percentage
5	4.756250	6.105120	78%
10	7.443250	10.365210	72%
15	9.523685	13.256840	72%
20	13.256470	16.524190	80%
25	1.2182180	15.965740	76%
30	23.965230	27.365210	88%
35	19.254150	22.654230	85%
40	41.426960	4.3520180	95%
45	32.427760	36.733240	95%
50	40.100680	41.300960	88%
55	47.684420	49.862270	97%
60	69.762950	78.467450	96%
65	82.236240	88.386430	89%
70	88.739410	96.520320	93%
75	73.096270	79.328690	92%
80	76.216940	94.359090	92%
85	91.283150	99.860260	81%
90	92.494110	93.051050	91%
95	89.090840	92.479740	96%
100	74.333890	106.840880	96%
Sum	49.463742	55.647355	88%

Table 5

The Operation Time for Case 3 (General Case)

No. / Dimension	Random Matrix	Solved by IIAA	Chance
5	100	21	21%
10	100	20	20%
15	100	12	12%
20	100	11	11%
25	100	13	13%
30	100	9	9%
35	100	6	6%
40	100	2	2%
45	100	11	11%
50	100	1	1%
55	100	1	1%
60	100	12	12%
65	100	7	7%
70	100	6	6%
75	100	3	3%
80	100	12	12%
85	100	4	4%
90	100	1	1%
95	100	1	1%
100	100	21	21%
Sum	2000	174	8.70%

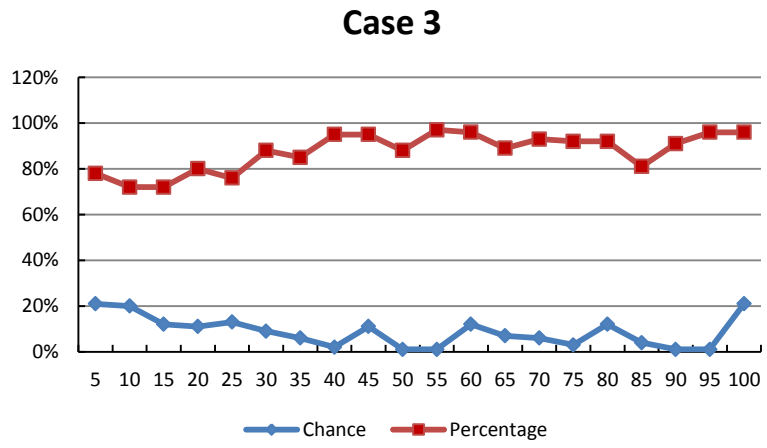


Figure 33：Chances and the percentage of general cases

Table 4 provides the average time of different dimension matrix which has been solved by Improved Incremental Assignment Algorithm and Incremental Assignment Algorithm for Case 2. Timing unit is millisecond. Table 5 presents the chances that a random matrix solved by IIAA

Figure 33 presents the trend line of chances that a random matrix solved by IIAA for different dimensions and the percentage of operation time of IAA

4.6 Performance Analysis

Table I and II show the typical data collected from the experiments stated previously. The average operation time of Incremental Assignment Algorithm (IAA) required by random matrix are linearly increased. It is possible for some particular smaller dimension matrix to take more time than a larger dimension matrix because the value distributions significantly affect the time needed in the relevant algorithm. The average running time of Improved Incremental Assignment Algorithm (IIAA) is stable regardless of the increase of the dimension. The difference average running time between IIAA and IAA becomes higher because of the difference of iteration. With the higher dimension, the iteration of IAA will go higher. But IIAA still have only one iteration no matter how high the dimension is.

In Case 1, IIAA saves up to 97% of operation time in high dimension matrix. In low dimension matrix, it will save 78% of running time on average. In Case 2, IIAA saves up to 97% of operation time in high dimension matrix. In low dimension matrix, it will save 71% of running time on average. The data express that Improved Incremental Assignment Algorithm will save much time when the dimension of the matrix is high.

In Case 3 (general cases), for all the 2000 random matrices, 8.7% matrices meet the conditions of Improved Incremental Assignment Algorithm, the rest of the matrices will go to Incremental Assignment Algorithm. In each dimension, when the matrix is solved by Improved Incremental Assignment Algorithm, the operation time will be 70% ~90% of the Incremental Assignment Algorithm.

Overall, when the matrix is solved by Improved Incremental Assignment Algorithm, the operation time will be faster than Incremental Assignment Algorithm.

4.7 Complexity

Because the algorithm involves the worst case is to executing the Incremental Assignment Algorithm, the computational complexity of the Improved Incremental Assignment Algorithm is same as the Incremental Assignment Algorithm which is $O(n^2)$.

Chapter 5

5 Incremental Group Role Assignment Problem

This chapter shows a way to solve Group Role Assignment Problem (GRAP) by using the Improved Incremental Assignment Algorithm (IIAA). This chapter will provide:

- The introduction of Incremental Group Role Assignment Problem (IGRAP),
- A real-world Problem,
- Solution to IGRAP,

5.1 Introduction

Incremental Group role assignment problem (IGRAP) initiates: Given a matrix which has m agents and n roles, where 0 and 1 mean no and yes, respectively. The roles are equally important. Its highest performance role assignments and the dual variables (u_i for rows and v_j for columns) are also given, determine the optimal solution of the group for the different situations below:

- extended with a new agent;
- extended with a new role;
- extended with a new agent and a new role at the same time;
- extended with a new agent and more than one role at the same time;
- extended with more than one agent and a new role at the same time;
- extended with more than one agent and more than one role at the same time.

Group Role Assignment problem can be solved by adjusting the number of agents and roles, the Role Group Role Assignment Problem (RGRAP) can be transferred to a General Assignment Problem (GAP). GAP has been solved by K-M algorithm. Since K-M algorithm solves the square matrix of GAP, which means the number of agents should be equal to that of roles. In the real world scenario, the assignments will not always meet this condition. Obviously, we need to transfer the input matrix into a square matrix. In order to form the square matrix, they duplicate rows or columns by using the Role Range Vector, Qualification Threshold Role Index Vector and other definitions and conditions to transfer RGRAP to GAP, as well as avoid the incorrect optimal result. After calling the K-M algorithm, the final assignment is produced based on the result. They provided the proofs and algorithms about how to transfer an RGRAP to a GAP and how a Weighted Group Role Assignment Problem (WGRAP) can be solved by the algorithm for an RGRAP [60].

The remainder of this chapter will be introduced as follows. 5.2 introduce the requirement of incremental group role assignment by a real world problem. 5.3 will talk about how to applying Improved Incremental Assignment Algorithm (IIAA) to solve IGRAP. 5.4 introduces other real world scenario about IGRAP can be solved by using IIAPA.

5.2 Real World Problem

In a walk-in clinic, the director of nursing department hopes to establish a system that can help him/her to assign different nurses to different departments. The departments will be included Registration Office, Reception Room, Consultation Room, Dressing Room, Therapeutic Department…Different department staff has different expertise.

A walk-in clinic has 20 nurses in total. There are four departments: Registration Office (one nurse), Dressing Room (four nurses), Consultation Room (three nurses), and Therapeutic Department (three nurses). This is the solution to assign the nurses to each department to get their best performance. Nurses are evaluated by their modes, emotions, professional certificate, work experience and GPA. The result will be shown as the weights in the matrix. Suppose that the director has the data shown in the table to express the evaluation values of nurses with respect to each department (rows represent nurses, and columns represent departments). The clinic nurses' performance is assumed as a simple sum of the selected nurses' performance on their designated departments. The optimal solution is as shown.

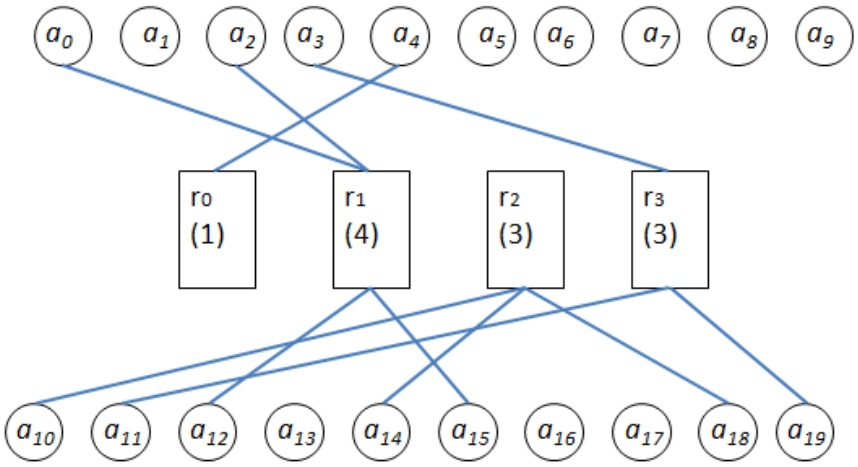


Figure 34： A clinic with 20 nurses and 4 departments

$$\begin{bmatrix} 0.65 & \underline{0.98} & 0.96 & 0.90 \\ 0.26 & 0.33 & 0.59 & 0.19 \\ 0.72 & \underline{0.61} & 0.19 & 0.63 \\ 0.06 & 0.48 & 0.43 & \underline{0.90} \\ \underline{0.87} & 0.35 & 0.06 & 0.25 \\ 0.72 & 0.15 & 0.28 & 0.01 \\ 0.33 & 0.59 & 0.37 & 0.67 \\ 0.75 & 0.59 & 0.25 & 0.45 \\ 0.12 & 0.10 & 0.01 & 0.51 \\ 0.84 & 0.13 & \underline{0.96} & 0.63 \\ 0.01 & 0.29 & 0.82 & 0.12 \\ 0.07 & 0.52 & 0.36 & \underline{0.95} \\ 0.97 & \underline{0.90} & 0.88 & 0.54 \\ 0.14 & 0.54 & 0.51 & 0.26 \\ 0.04 & 0.03 & \underline{0.83} & 0.70 \\ 0.44 & \underline{0.70} & 0.16 & 0.39 \\ 0.12 & 0.48 & 0.04 & 0.76 \\ 0.30 & 0.14 & 0.52 & 0.08 \\ 0.91 & 0.50 & \underline{0.96} & 0.21 \\ 0.53 & 0.06 & 0.85 & \underline{0.85} \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b)

Figure 35: Evaluation values of agents and roles and the assignment matrix

Case 1:

During the graduation season, nurses graduated from nursing school and are looking for jobs. Because the number of patients increases, the clinic expanded its scale to open a new department – Reception Room. The director of nursing department need to assign the new graduates to each department (combine the old and the new) to achieve their optimal performance in the clinic. This is exactly an Incremental Group Role Assignment Problem that can be solved by applying the Improved Incremental Assignment Algorithm in the real world in some cases.

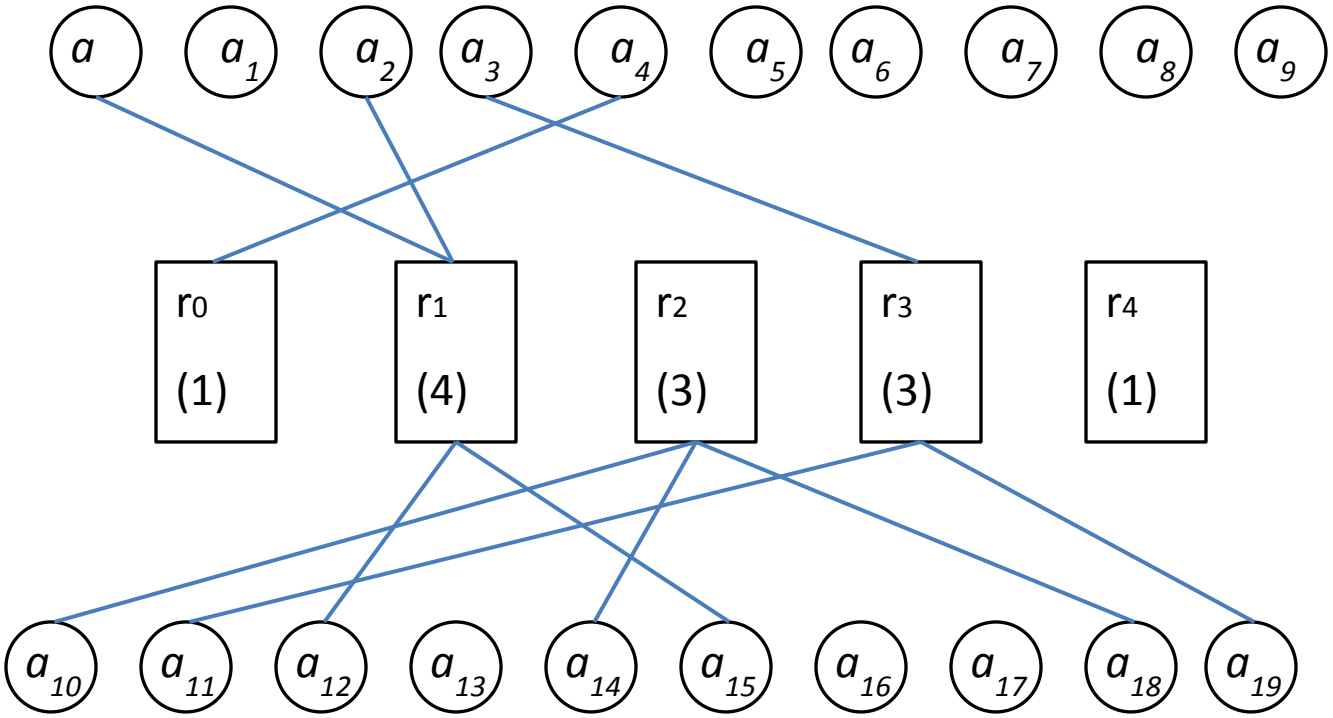


Figure 36: A clinic adding a new department

0.65	0.98	0.96	0.90	0.30
0.26	0.33	0.59	0.19	0.36
0.72	0.61	0.19	0.63	0.43
0.06	0.48	0.43	0.90	0.55
0.87	0.35	0.06	0.25	0.32
0.72	0.15	0.28	0.01	0.29
0.33	0.59	0.37	0.67	0.89
0.75	0.59	0.25	0.45	0.51
0.12	0.10	0.01	0.51	0.20
0.84	0.13	0.96	0.63	0.03
0.01	0.29	0.82	0.12	0.32
0.07	0.52	0.36	0.95	0.13
0.97	0.90	0.88	0.54	0.20
0.14	0.54	0.51	0.26	0.37
0.04	0.03	0.83	0.70	0.16
0.44	0.70	0.16	0.39	0.45
0.12	0.48	0.04	0.76	0.35
0.30	0.14	0.52	0.08	0.26
0.91	0.50	0.96	0.21	0.61
0.53	0.06	0.85	0.85	0.52

(a)

0	1	0	0	0
0	0	0	0	0
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	1	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	1	0

(b)

Figure 37: Evaluation values of agents and roles and the assignment matrix

The clinic nurses' performance is assumed as a simple sum of the selected nurses' performance on their designated departments. By using GRAP, the optimal solution is $r_0 = \{a_4\}$, $r_1 = \{a_0, a_2, a_{12}, a_{15}\}$, $r_2 = \{a_{10}, a_{14}, a_{18}\}$, $r_3 = \{a_3, a_{11}, a_{19}\}$, $r_4 = \{a_6\}$. This assignment obtains the best total group performance of 10.4. The optimal solution is shown in Figure 35(a) the number with the line under, in Figure 35(b) as a matrix, and in Figure 36 as a graph.

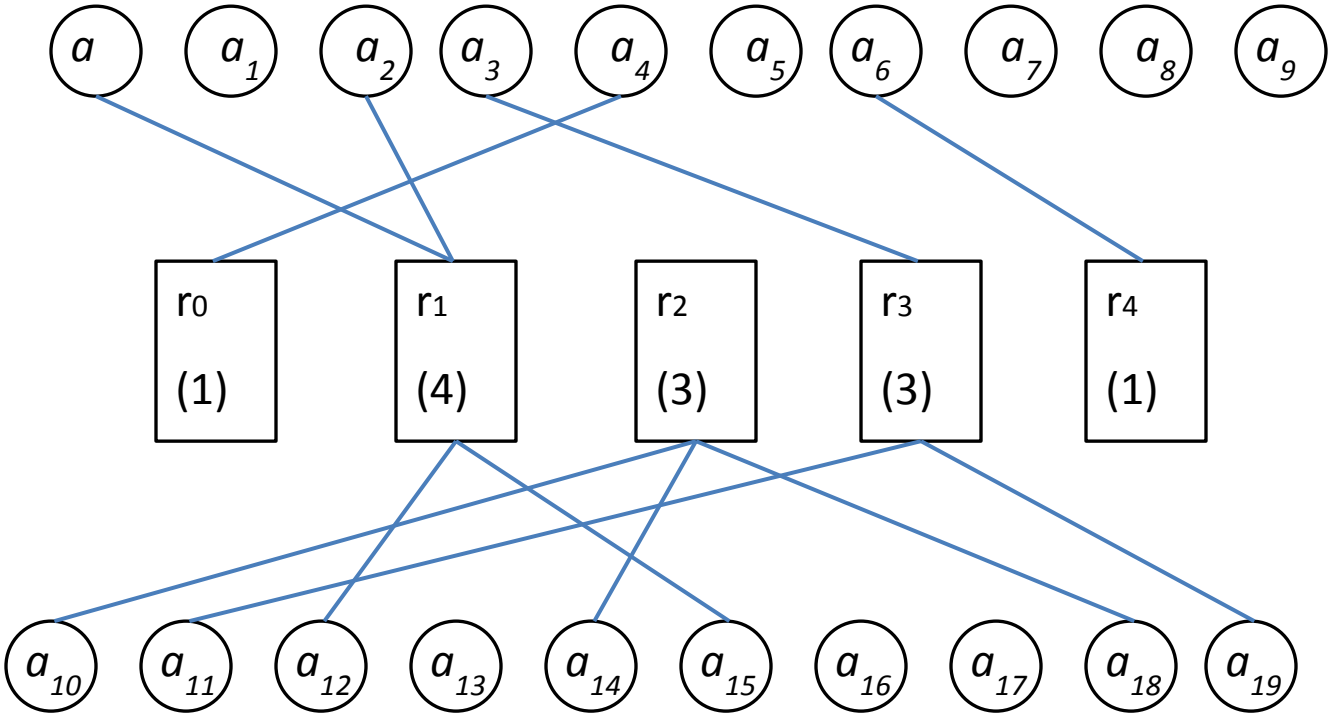


Figure 38: Optimal solution

Case 2:

A new nurse (agent) joins the clinic, which makes the group of 21 nurses ($a_0 \sim a_{21}$). Figure 37 shows the 21 nurses and the 4 departments.

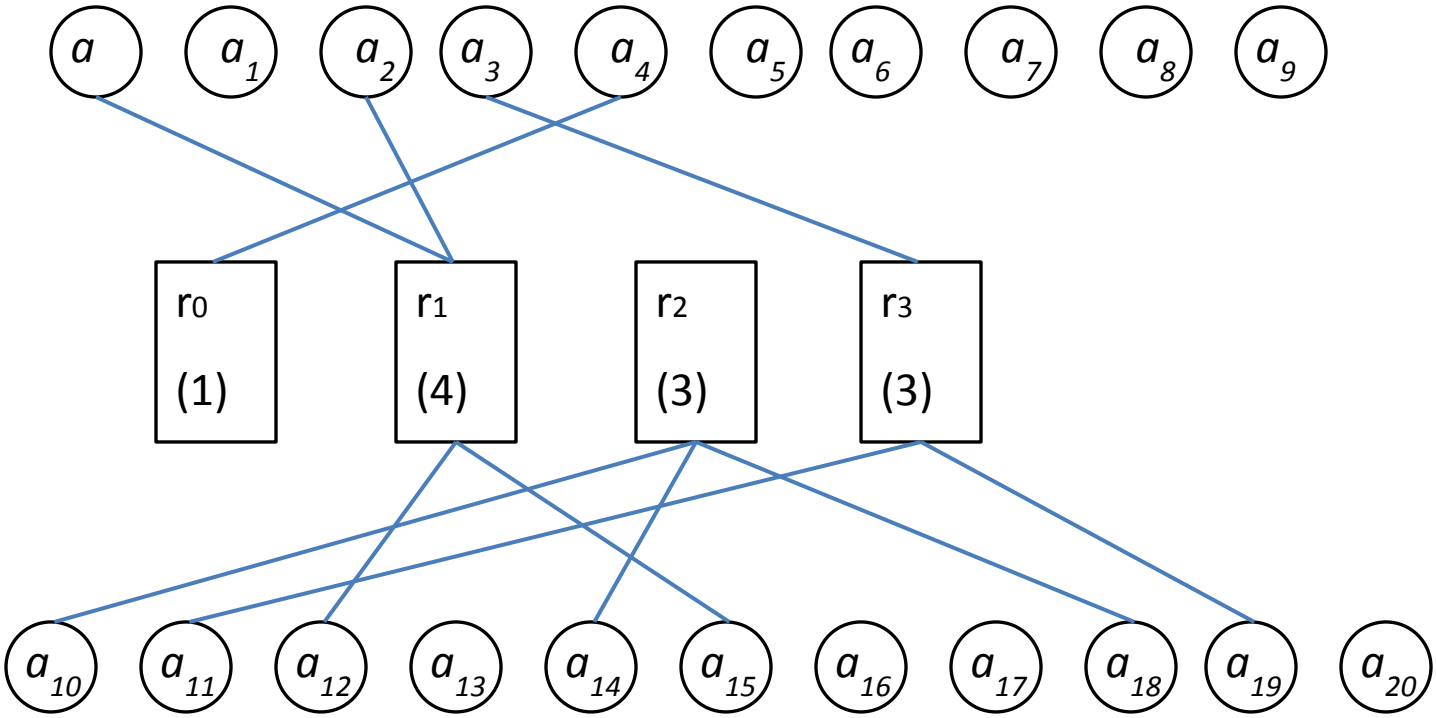


Figure 39: A new nurse joins the clinic

0.65	0.98	0.96	0.90
0.26	0.33	0.59	0.19
0.72	0.61	0.19	0.63
0.06	0.48	0.43	0.90
0.87	0.35	0.06	0.25
0.72	0.15	0.28	0.01
0.33	0.59	0.37	0.67
0.75	0.59	0.25	0.45
0.12	0.10	0.01	0.51
0.84	0.13	0.96	0.63
0.01	0.29	0.82	0.12
0.07	0.52	0.36	0.95
0.97	0.90	0.88	0.54
0.14	0.54	0.51	0.26
0.04	0.03	0.83	0.70
0.44	0.70	0.16	0.39
0.12	0.48	0.04	0.76
0.30	0.14	0.52	0.08
0.91	0.50	0.96	0.21
0.53	0.06	0.85	0.85
0.25	0.78	0.09	0.53

(a)

0	1	0	0
0	0	0	0
0	0	0	0
0	0	0	1
1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	1
0	1	0	0
0	0	0	0
0	0	1	0
0	1	0	0
0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0

(b)

Figure 40: Evaluation values of nurses and departments and the assignment matrix

By using GRAP, the optimal solution is $r_0 = \{a_4\}$, $r_1 = \{a_0,\ a_{12}, a_{15}, a_{20}\}$, $r_2 = \{a_{10}, a_{14}, a_{18}\}$, $r_3 = \{a_3, a_{11}, a_{19}\}$. This assignment obtains the best total group performance of 9.68. The optimal solution is shown in Figure 38(a) the number with the line under, in Figure 38(b) as a matrix, and in Figure 39 as a graph.

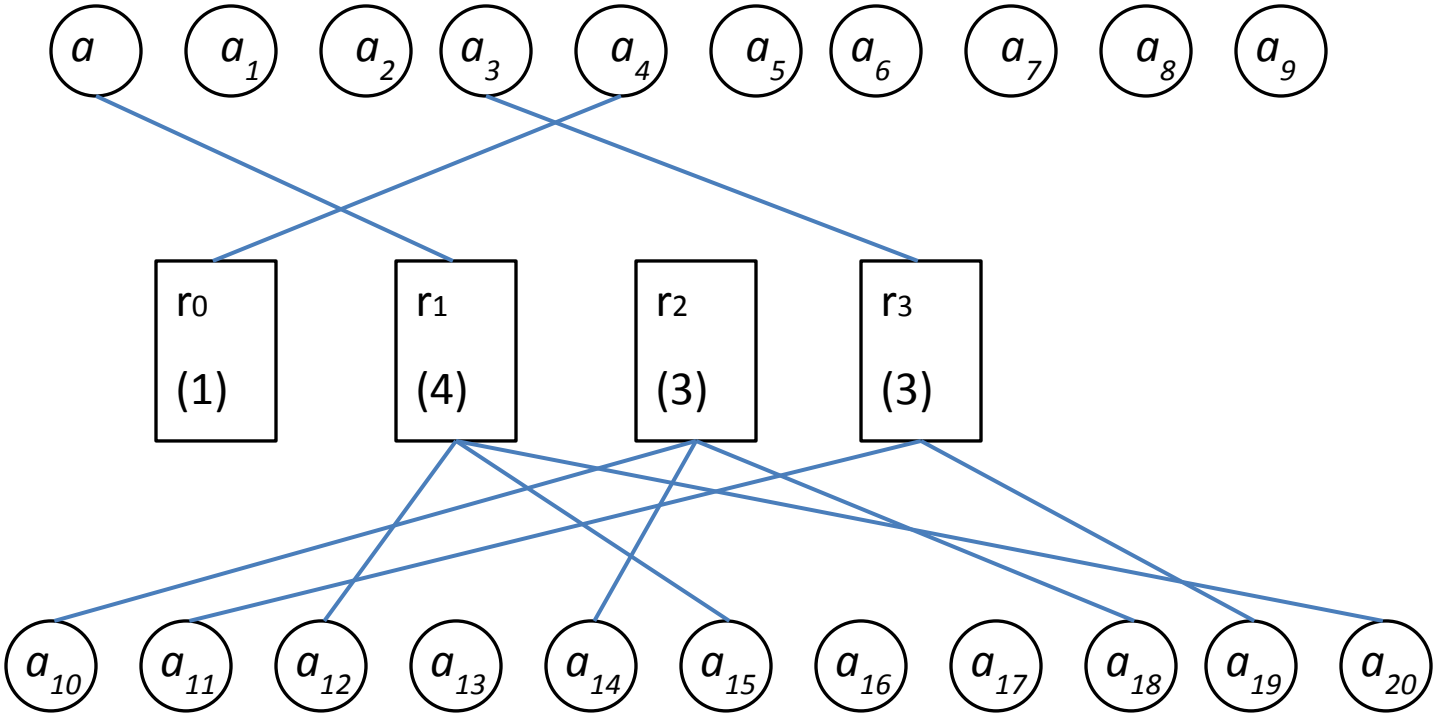


Figure 41: Optimal solution

Case 3:

The director wants to add a new department and a new nurse joins the clinic. The department has one nurse in it. Figure 40 shows the 21 nurses and the 5 departments.

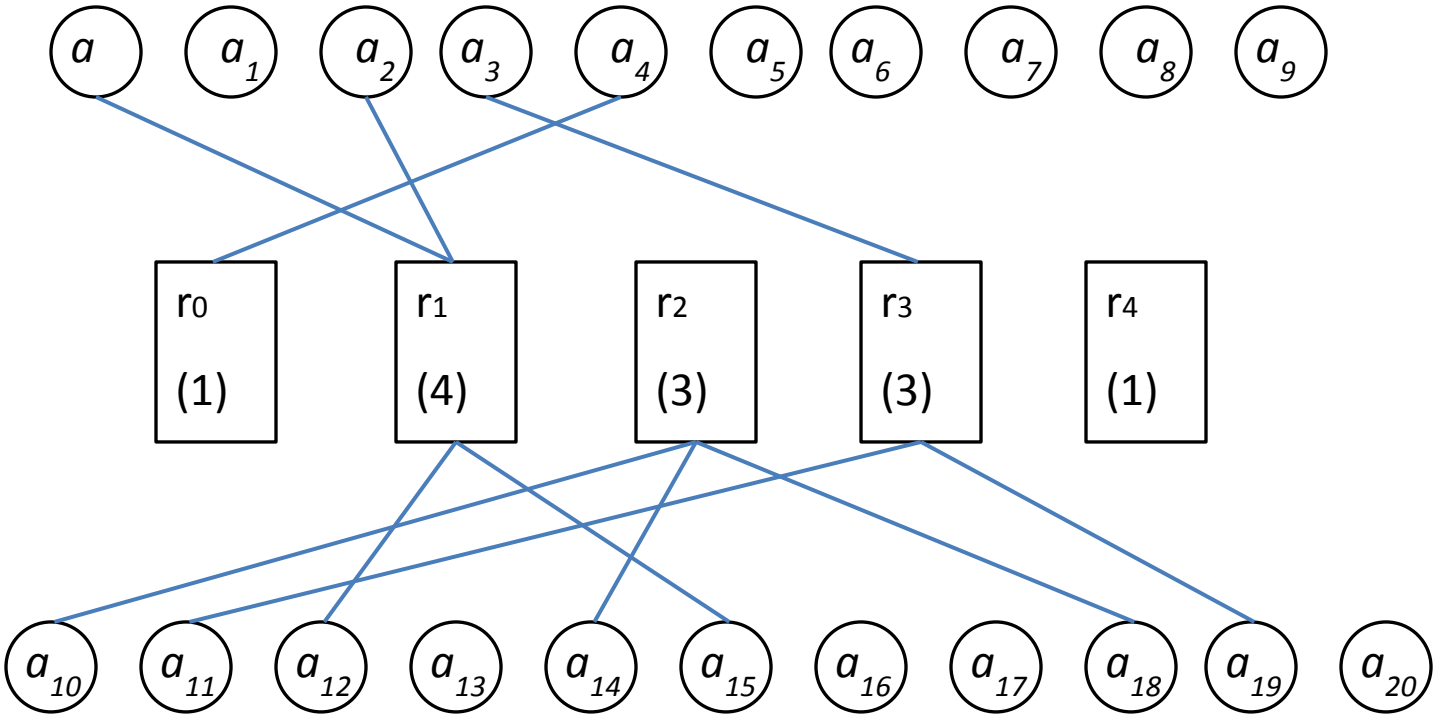


Figure 42: A new department and a new nurse join the clinic

$$\begin{bmatrix}
0.65 & \underline{0.98} & 0.96 & 0.90 & 0.30 \\
0.26 & 0.33 & 0.59 & 0.19 & 0.36 \\
0.72 & 0.61 & 0.19 & 0.63 & 0.43 \\
0.06 & 0.48 & 0.43 & \underline{0.90} & 0.55 \\
\underline{0.87} & 0.35 & 0.06 & 0.25 & 0.32 \\
0.72 & 0.15 & 0.28 & 0.01 & 0.29 \\
0.33 & 0.59 & 0.37 & 0.67 & \underline{0.89} \\
0.75 & 0.59 & 0.25 & 0.45 & 0.51 \\
0.12 & 0.10 & 0.01 & 0.51 & 0.20 \\
0.84 & 0.13 & \underline{0.96} & 0.63 & 0.03 \\
0.01 & 0.29 & 0.82 & 0.12 & 0.32 \\
0.07 & 0.52 & 0.36 & \underline{0.95} & 0.13 \\
0.97 & \underline{0.90} & 0.88 & 0.54 & 0.20 \\
0.14 & 0.54 & 0.51 & 0.26 & 0.37 \\
0.04 & 0.03 & \underline{0.83} & 0.70 & 0.16 \\
0.44 & \underline{0.70} & 0.16 & 0.39 & 0.45 \\
0.12 & 0.48 & 0.04 & 0.76 & 0.35 \\
0.30 & 0.14 & 0.52 & 0.08 & 0.26 \\
0.91 & 0.50 & \underline{0.96} & 0.21 & 0.61 \\
0.53 & 0.06 & 0.85 & \underline{0.85} & 0.52 \\
0.25 & \underline{0.78} & 0.09 & 0.53 & 0.33
\end{bmatrix}
\qquad
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

(a) (b)

Figure 43: Evaluation values of nurses and departments and the assignment matrix

By using GRAP, the optimal solution is $r_0 = \{a_4\}$, $r_1 = \{a_0,\ a_{12}, a_{15}, a_{20}\}$, $r_2 = \{a_{10}, a_{14}, a_{18}\}$, $r_3 = \{a_3, a_{11}, a_{19}\}$, $r_4 = \{a_6\}$. This assignment obtains the best total group performance of 10.57. The optimal solution is shown in Figure 41(a) the number with the line under, in Figure 41(b) as a matrix, and in Figure 42 as a graph.

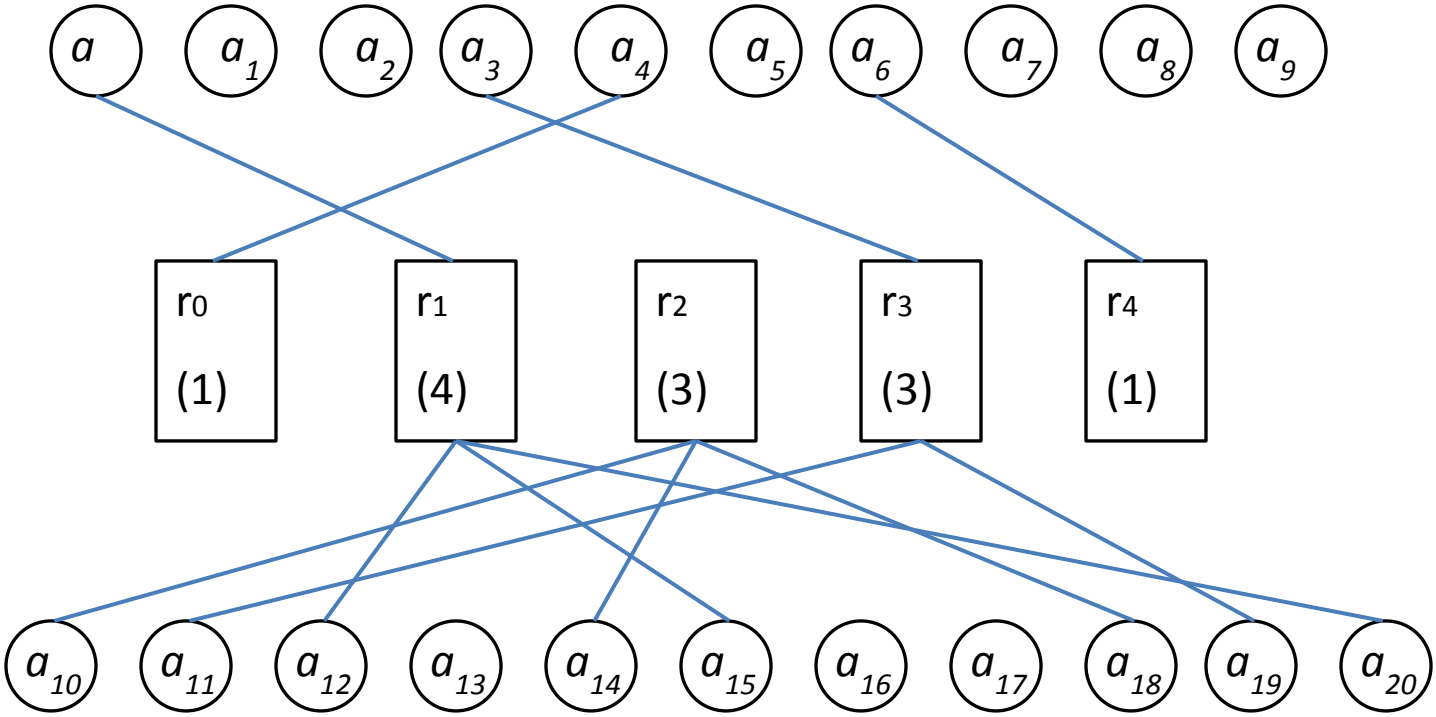


Figure 44: Optimal solution

5.3 Solution to IGRAP

5.3.1 Concepts

To clarify the Incremental Group Role Assignment (IGRAP), we need to state some concepts first. In formalizing IGRAPs, m expresses the size of the agent, and n expresses the size of the role. For example, in the soccer team, m is 20 players, n is four roles as goalkeeper, backs, midfields, forwards and backup.

Role Range Vector: A role range vector is a vector of the lower ranges of roles. The role range vector is denoted as $L[j] \in N$, where N is the set of natural numbers and $0 \leq j < n$.

For example, $L = [1, 4, 3, 3, 1]$ for the soccer team in Figure 34.

Qualification Matrix: The qualification matrix is an $m \times n$ matrix of values in $[0, 1]$,where $Q[i, j]$ expresses the qualification value of agent i for role j. It is denoted as $Q[i, j] \in [0, 1](0 \leq i < m; 0 \leq j < n)$.

For example, Figure 35(a) shows a qualification matrix for the soccer team.

Role Assignment Matrix: A role assignment matrix is an $m \times n$ matrix of values in $\{0, 1\}$. If $T[i, j] = 1$, agent i is assigned to role j, and agent i is called an assigned agent. It is denoted as $T[i, j] \in \{0, 1\}(0 \leq i < m; 0 \leq j < n)$.

For example, Figure 35(b) shows an role assignment matrix for the soccer team.

Group Qualification: A group qualification is defined as the sum of the assigned agents' qualifications, i.e. $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j]$.

For example, the group qualification of Figure 35 is 10.4.

5.3.2 Solution to IGRAP

Given a matrix that has i agents and j roles, where 0 and 1 mean no and yes, respectively. The roles are equally important. The maximization assignment and the dual variables (u_i for rows and v_i for columns) have already been given. One condition is the total agent number (m) should be larger than the required numbers, $m \geq \sum_{j=0}^{n-1} L[j]$.

There are three situations we need to consider:

- $m > n$: The number of agents (m) is higher than the number of roles (n). And each role can have more than one agent which is shown in Figure 10.
- $m = n$: The number of agents (m) equal to the number of roles (n). And each role only has one agent.
- $m < n$: The number of agents (m) is lower than the number of roles (n). And each role can have more than one agent. This situation cannot happen because of the condition.

	Y_1	Y_2	Y_3	α_i
X_1	0.36	0.76	0.72	0.43
X_2	0.93	0.59	0.24	0.53
X_3	0.06	0.46	0.69	0.69
X_4	0.40	0.10	0.74	0.74
X_5	0.23	0.75	0.24	0.75
X_6	0.21	0.77	0.24	0.77
β_i	0.4	0.3	0	

Figure 45: Matrix with optimal solution

Situation 1:

Case 1(1):

The matrix is extended with a new column which corresponding a role. Figure 11 shows an example of the extended matrix. This matrix has 6 rows corresponding to agents. Its role range vector is $L = [1, 2, 1, 1]$. In Figure 11, $m > \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	0.36	0.76	0.72	0.32	0.43
X_2	0.93	0.59	0.24	0.43	0.53
X_3	0.06	0.46	0.69	0.59	0.69
X_4	0.40	0.10	0.74	0.25	0.74
X_5	0.23	0.75	0.24	0.16	0.45

X_6	0.21	0.77	0.24	0.26	0.47
β_i	0.4	0.3	0		

Figure 46: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{2}, \mathbf{1}, \mathbf{1}]$

Subtracting all the rows with chosen agents, then we get the matrix in Figure 45. By comparing the variable in the adding column, we can get the maximum result in the column Y_5. The optimal solution is 3.79.

	Y_1	Y_2	Y_3	Y_4	α_i
X_3	0.06	0.46	0.69	0.59	0.69
X_5	0.23	0.75	0.24	0.16	0.45
β_i	0.4	0.3	0		

Figure 47: Matrix after subtracting

Case 1(2):

Figure 46 shows another example of the extended matrix. This matrix has 6 rows corresponding to agents. Its role range vector is $L = [1, 2, 1, 2]$. In this case, $m = \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	0.36	0.76	0.72	0.32	0.43
X_2	0.93	0.59	0.24	0.43	0.53
X_3	0.06	0.46	0.69	0.59	0.69
X_4	0.40	0.10	0.74	0.25	0.74
X_5	0.23	0.75	0.24	0.16	0.45

X_6	0.21	0.77	0.24	0.26	0.47
β_i	0.4	0.3	0		

Figure 48: Matrix extended with a new column, $\boldsymbol{L = [1, 2, 1, 2]}$

Subtracting all the rows with chosen agents, then we get the matrix in Figure 47. This will be easier that there are two left agents and the goal is to choose two agents. Both of the left agents need to be chosen. The maximum result in the column Y_5. The optimal solution is 3.95.

	Y_1	Y_2	Y_3	Y_4	α_i
X_3	0.06	0.46	0.69	0.59	0.69
X_5	0.23	0.75	0.24	0.16	0.45
β_i	0.4	0.3	0		

Figure 49: Matrix after subtracting

Case 2(1):

The matrix is extended with a new row corresponding to an agent. Figure 48 shows an example of the extended matrix. This matrix has 7 rows corresponding to agents. Its role range vector is $L = [1, 2, 1]$. In Figure 14, $m > \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	α_i
X_1	0.36	0.76	0.72	0.43
X_2	0.93	0.59	0.24	0.53
X_3	0.06	0.46	0.69	0.69
X_4	0.40	0.10	0.74	0.74
X_5	0.23	0.75	0.24	0.45
X_6	0.21	0.77	0.24	0.47

X_7	0.65	0.43	0.88	
β_i	0.4	0.3	0	

Figure 50: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{2}, \mathbf{1}]$

In this case, we need to apply the GRAP to get the maximum result which is shown in Figure 49. The optimal solution is 3.94.

	Y_1	Y_2	Y_3
X_1	0.36	0.76	0.72
X_2	0.93	0.59	0.24
X_3	0.06	0.46	0.69
X_4	0.40	0.10	0.74
X_5	0.23	0.75	0.24
X_6	0.21	0.77	0.24
X_7	0.65	0.43	0.88

Figure 51: Optimal solution

Case 2(2):

The matrix is extended with more than one row corresponding to an agent. Figure 50 shows an example of the extended matrix with two agents. This matrix has 8 rows corresponding to agents. Its role range vector is $L = [1, 2, 1]$. In this case, $m > \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	α_i
X_1	0.36	0.76	0.72	0.43
X_2	0.93	0.59	0.24	0.53
X_3	0.06	0.46	0.69	0.69
X_4	0.40	0.10	0.74	0.74
X_5	0.23	0.75	0.24	0.45

X_6	0.21	0.77	0.24	0.47
X_7	0.65	0.43	0.88	
X_8	0.25	0.80	0.13	
β_i	0.4	0.3	0	

Figure 52: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{2}, \mathbf{1}]$

In this case, we need to apply the GRAP to get the maximum result which is shown in Figure 51. The optimal solution is 3.94.

	Y_1	Y_2	Y_3
X_1	0.36	0.76	0.72
X_2	0.93	0.59	0.24
X_3	0.06	0.46	0.69
X_4	0.40	0.10	0.74
X_5	0.23	0.75	0.24
X_6	0.21	0.77	0.24
X_7	0.65	0.43	0.88

Figure 53: Optimal solution

Case 3:

The matrix is extended with a new column corresponding a role and a new row corresponding to an agent at the same time. This is a new instance of the incremental problem which adds an agent and a role at the same time. Figure 19 shows an example of the extended matrix. This matrix has 6 rows corresponding to agents. Its role range vector is $L = [1, 2, 1, 1]$. In Figure 52, $m > \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	0.36	0.76	0.72	0.32	0.43
X_2	0.93	0.59	0.24	0.43	0.53
X_3	0.06	0.46	0.69	0.59	0.69
X_4	0.40	0.10	0.74	0.25	0.74
X_5	0.23	0.75	0.24	0.16	0.45
X_6	0.21	0.77	0.24	0.26	0.47
X_7	0.65	0.43	0.88	0.92	
β_i	0.4	0.3	0		

Figure 54: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{2}, \mathbf{1}, \mathbf{1}]$

By applying the GRAP, we can get the maximum result which is shown in Figure 53. The optimal solution is 4.12.

	Y_1	Y_2	Y_3	Y_4
X_1	0.36	0.76	0.72	0.32
X_2	0.93	0.59	0.24	0.43
X_3	0.06	0.46	0.69	0.59
X_4	0.40	0.10	0.74	0.25
X_5	0.23	0.75	0.24	0.16
X_6	0.21	0.77	0.24	0.26
X_7	0.65	0.43	0.88	0.92

Figure 55: Optimal solution

Situation 2:

$m = n$: The number of agents (m) equal to the number of roles (n). And each role only has one agent which is shown in Figure 54.

	Y_1	Y_2	Y_3	Y_4	α_i
X_1	0.36	0.76	0.72	0.32	0.43
X_2	0.93	0.59	0.24	0.43	0.53
X_3	0.40	0.10	0.74	0.25	0.74
X_4	0.21	0.39	0.24	0.77	0.77
β_i	0.4	0.3	0	0	

Figure 56: Matrix with optimal solution ($\boldsymbol{L} = [\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1}]$)

Case 1:

The matrix is extended with a new column corresponding to a role and a new row corresponding to an agent at the same time. This is typical incremental problem which can be solved by applying the Improved Incremental Assignment Algorithm. Figure 55 shows an example of the extended matrix. This matrix has 6 rows corresponding to agents. Its role range vector is $L = [1, 1, 1, 1]$. In Figure 22, $m = \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	Y_4	Y_5	α_i
X_1	0.36	0.76	0.72	0.32	0.53	0.43
X_2	0.93	0.59	0.24	0.43	0.26	0.53
X_3	0.40	0.10	0.74	0.25	0.15	0.74
X_4	0.21	0.39	0.24	0.77	0.34	0.77
X_5	0.36	0.83	0.63	0.45	0.96	
β_i	0.4	0.3	0	0		

Figure 57: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1}]$

By applying the IIAP, we can get the maximum result which is shown in Figure 56. The optimal solution is 4.16.

	Y_1	Y_2	Y_3	Y_4	Y_5
X_1	0.36	0.76	0.72	0.32	0.53
X_2	0.93	0.59	0.24	0.43	0.26
X_3	0.40	0.10	0.74	0.25	0.15
X_4	0.21	0.39	0.24	0.77	0.34
X_5	0.36	0.83	0.63	0.45	0.96

Figure 58: Optimal solution

Obviously, instead of applying the Kuhn-Munkres Algorithm which complexity is $O(n^3)$ to solve IGRAP, Improved Incremental Assignment Algorithm will save much time and space to solve the Incremental Group Role Assignment Problem (IGRAP).

Case 2:

The matrix is extended with more than one columns (k) corresponding to the roles and more than one rows (t) corresponding the agents at the same time. In this case, $t > k$. This is new type of incremental group role assignment problem. Figure 57 shows an example of the extended matrix. Its role range vector is $L = [1, 1, 1, 2, 1]$. In Figure 24, $m > \sum_{j=0}^{n-1} L[j]$ which meets the condition.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	α_i
X_1	0.36	0.76	0.72	0.32	0.53	0.66	0.43
X_2	0.93	0.59	0.24	0.43	0.26	0.59	0.53
X_3	0.40	0.10	0.74	0.25	0.15	0.10	0.74
X_4	0.21	0.39	0.24	0.77	0.34	0.39	0.77
X_5	0.36	0.53	0.63	0.45	0.96	0.83	
X_6	0.36	0.50	0.26	0.50	0.58	0.76	
X_7	0.14	0.63	0.29	0.62	0.90	0.45	
X_8	0.25	0.60	0.24	0.37	0.46	0.86	
β_i	0.4	0.3	0	0			

Figure 59: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{1}]$

By applying the GRAP, we can get the maximum result which is shown in Figure 57. The optimal solution is 5.92.

Case 3:

The matrix is extended with more than one columns (k) corresponding to the roles and more than one rows (t) corresponding to the agents at the same time. In this case, $t < k$. This is not incremental group role assignment problem because $m < \sum_{j=0}^{n-1} L[j]$ which does not meet the condition. Figure 58 shows an example of the extended matrix. Its role range vector is $L = [1, 1, 1, 2, 1]$.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	α_i
X_1	0.36	0.76	0.72	0.32	0.53	0.66	0.43
X_2	0.93	0.59	0.24	0.43	0.26	0.59	0.53
X_3	0.40	0.10	0.74	0.25	0.15	0.10	0.74
X_4	0.21	0.39	0.24	0.77	0.34	0.39	0.77
X_5	0.36	0.83	0.63	0.45	0.96	0.83	
β_i	0.4	0.3	0	0			

Figure 60: Matrix extended with a new column, $\boldsymbol{L} = [\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{1}]$

Chapter 6

6 Conclusions

In this thesis, an improved algorithm has been proposed to solve Incremental Assignment Problem. From the view of the overall program, the algorithm improves a lot with the running time of operation. From finding the largest difference of the row and the column, with the use of exchange with its maximum weighted matching, the problem has been reduced. Consequently, the step of iteration can be reduced largely.

The Improved Incremental Assignment Algorithm also has good performance when solve the Incremental Group Role Assignment Problem. If the case meets the conditions of Case 1 and Case 2, instead of the complexity of $O(n^2)$, the problem can be solved by only using one iteration. The performance of experiments has shown the advantage of our algorithm.

From the perspective of functions, our solution directly provides an improved way to solving the Incremental Assignment Problem. This solution can also be used to improve the Group Role Assignment Problem when there is a pair of new agent and new role (which has one agent in the group) adding to the matrix. Not only the operation time will be saved, but also the occupied space will be reduced.

The computational complexity of our algorithm is $O(n^2)$, because the most complicated case for our algorithm will go to Incremental Assignment Algorithm. As the complexity of Incremental Assignment Algorithm is $O(n^2)$, our algorithms are also at the same level.

Our solution has many advantages, but it still necessary to point out the disadvantages. In general cases, about 8.7% of the 2000 random matrices meet the conditions of Improved Incremental Assignment Algorithm, the rest of the matrix will go to Incremental Assignment Algorithm. The chance is limited which need to be improved in the future.

References

[1] F. Syakinah, S. Abdul-Rahman, and R. Abd Rahman. An Assignment Problem and Its Application in Education Domain: A Review and Potential Path. Research article. Advances in Operations Research, 2018. avail: https://www.hindawi.com/journals/aor/2018/8958393/

[2] H. Basirzadeh. Ones assignment method for solving assignment problems, Applied Mathematical Sciences, vol. 6, no. 45-48, pp. 2345–2355, 2012.

[3] R. E. Burkard. Selected Topics on Assignment Problems. Discrete Applied Mathematics 123, no. 1–3 (November 2002): 257–302.

[4] R. E. Burkard, M. Dell'Amico, S. Martello. Assignment Problems. Society for Industrial and Applied Mathematics, pp. 35-144, 2009.

[5] D.A Grundel and P.M. Pardalos. Test problem generator for the multidimensional assignment problem. Computational Opt. Appl., 30:133-146,2005. (Cited on p. 324)

[6] J. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput., 2:225-231,1973.

[7] J.F. Benders. Partitioning procedures for solving mixed variables programming problems. Numerische Mathematik, 4:238-252, 1962. (Cited on p. 254)

[8] S.E. Alm and G.B. Sorkin. Exact expectations and distributions for the random assignment problem. Combin. Probab. Comput., 11:217-248, 2002. (Cited on p. 147.)

[9] J. Chakrapani and J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. Ann. Oper. Res., 41:327-341, 1993. (Cited on p.266)

[10] M. Mucha, P. Sankowski. Maximum Matchings via Gaussian Elimination. Proc. 45th IEEE Symp. Foundations of Computer Science, pp. 248–255, 2004.

[11] V. S. Árgilán, J. Balogh, and A. Tóth. The Basic Problem of Vehicle Scheduling Can Be Solved by Maximum Bipartite Matching. Proceedings of the 9th International Conference on Applied Informatics Eger, Hungary, Vil. 2. pp. 209-218, 2014.

[12] S. Bunte, and N. Kliewer. An Overview on Vehicle Scheduling Models. Public Transport 1, no. 4: 299–317, November 2009.

[13] R. E. Burkard. Time-Slot Assignment for TDMA-Systems. Computing 35, no. 2: 99–112, June 1985.

[14] B. Neng, Zur Erstellung von optimalen Triebfahrzeugumläufen,*Zeitschrift für Operations Research Ser. A–B* 25, 159–185,1981.

[15] R.E.Burkard, E. Çela. Linear assignment problems and extensions, in: D.Z. Du and P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Volume A, pages 75-149. Kluwer Academic Publishers, Dordrecht, 1999.

[16] M. Dell'Amico, S. Martello. Linear Assignment, in: M. Dell'Amico, F. MaQoli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, pages 355-371. John Wiley and Sons, Chichester, 1997.

[17] Y. Lee, J.B. Orlin. On very large scale assignment problems. In: W.W. Hager, D.W. Hearn, P.M. Pardalos (Eds.), Large Scale Optimization: State of the Art, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp.206–244,1994.

[18] R.M. Karp. An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$, Networks, 10:143–152, 1980.

[19] G. A. Korsah, A. Stentz and M. B. Dias. The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs. Tech. Report, CMU-RI-TR-07-27, Robotics Institute, Carnegie Mellon University, July, 2007.

[20] R. Jonker and A.T. Volgenant. Improving the Hungarian assignment algorithm. Oper. Res. Lett., 5:171-175,1986.

[21]L.F. McGinnis. Implementation and testing of a primal-dual algorithm for the assignment problem. Oper. Res., 31:277-291,1998.

[22] G. Carpaneto and P. Toth. Algorithm 648: Solution of the assignment problem. ACM Transactions on mathematical Software, 6:104-111, 1980.

[23] G. Carpaneto, S. Martello, and P. Toth. Algorithms and codes for the assignment problem. Fortan codes for network Optimization, volume 13 of Ann. Oper. Res., pages 193-223. Baltzer, Basel, 1988.

[24] F. Glover, D. Karney, and D. Klingman. Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems. Networks, 4:191-212, 1974.

[25] E.A. Dinic and M.A. Kronrod. An algorithm for the solution of the assignment problem. Sov. Math. Dokl., 10:1324-1326, 1969.

[26] A.J. Hoffman and H.M. Markowitz. A note on shortest paths, assignment, and transportation problems. Naval Res. Log. Quart., 10:375-379,1963.

[27] N. Tomizawa. On some techniques useful for solution of transportation network problems. Networks, 1:173-194,1971.

[28] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM, 19:248-264, 1972.

[29] G.B. Dantzig. Linear Programming and Extensions. Princeton University Press, Princeton, NJ, 1963.

[30] B. Gavish, P.Schweitzer, and E. Shlifer. The zero pivot phenomenon in transportation and assignment problems and its computational implications. Math. Program., 12:226-240, 1977.

[31] D.R. Fulkerson, I. Glicksberg, and O. Gross. A production line assignment problem. Tech. Rep. RM-1102, The Rand Corporation, Santa Monica, CA, 1953.

[32] R.E. Burkard, W.Hahn, and U. Zimmermann. An algebraic approach to assignment problems. Math. Program., 12:318-327,1977.

[33] G. Grygiel. Algebraic $\sum_k -$assignment problems. Control and Cybernetics, 10:155-165, 1981.

[34] S. Martelllo, W.R. Pulleyblank, P. Toth, and D. de Werra. Balanced optimization problems. Oper. Res. Lett., 3:275-278, 1984.

[35] T.C. Koopmans and M.J. Beckmann. Assignment problems and the location of economic activities. Econometrica, 25:53-76, 1957.

[36] W.P. Pierskalla. The multidimensional assignment problem. Oper. Res., 16:422-431, 1968.

[37] A.B. Poore. Multidimensional assignment and multitarget tracking. In Partitioning Data Sets, volumn 19 of DIMACS Series, Pages 169-196. American Mathematical Society, 1995.

[38] Community, Competitive Programming, Competitive Programming Tutorials, Assignment Problem and Hungarian Algorithm. avail: https://www.topcoder.com/community/competitive-programming/tutorials/assignment-problem-and-hungarian-algorithm/

[39] H. Zhu. Separating Design from Implementations: Role-Based Software Development, In Proc. of the 5th IEEE International Conference on Cognitive Informatics, ICCI'06, 17–19, n.d.

[40] H. Zhu, M. Zhou and R. Alkins, Group Role Assignment via a Kuhn–Munkres Algorithm-Based Solution, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 42, no. 3, pp. 739-750, May 2012.

[41] H. Shamakhai. The 0-1 Multiple Knapsack Problem. avail: https://pdfs.semanticscholar.org/703c/cf2cd4244c9521d95c70e37a5ad3e1eac562.pdf

[42] G. T. Ross, and R. M. Soland. A branch and bound algorithm for the generalized assignment problem. Mathematical programming, 8(1), 91-103,1975.

[43] S. Martello, and P. Toth. The 0-1 knapsack problem. In N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds). Combinatorial Optimization, Wiley, Chichester, 237-279, 1979.

[44] S. Martello, and P. Toth. Algorithms for knapsack problems. In S. Martello, G. Laporte, M. Minoux C. Ribeiro (eds), Surveys in Combinatorial Optimization, Annals of Discrete Mathematics 31, North-Holland, Amsterdam, 213-257, 1987.

[45] M. L. Fisher, and R. Jaikumar, and L. N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. Management Science, 32(9), 1095-1103, 1986.

[46] M. Guignard, and M. B. Rosenwein. Technical Note—An Improved Dual Based Algorithm for the Generalized Assignment Problem. Operations Research, 37(4), 658-663, 1989.

[47] A. Drexl. Scheduling of project networks by job assignment. Management Science, 37(12), 1590-1602, 1991.

[48] R. M. Nauss. Solving the generalized assignment problem: An optimizing and heuristic approach. Informs Journal on Computing, 15(3), 249-266, 2003.

[49] R.M. Nauss. The elastic generalized assignment problem. J Oper Res Soc 55:1333–1341,2005.

[50] M. Posta, J.A. Ferland, and P. Michelon. An exact method with variable fixing for solving the generalized assignment problem. Comp. Opt. and Appl., 52, 629-644, 2012.

[51] M. Savelsbergh. A branch and price algorithm for the generalized assignment problem. Operations research, 45(6), 831-841, 1997.

[52] S. Martello, P. Toth. An algorithm for the generalized assignment problem. In: Brans JP (ed). Operational Research ’81, 9th IFORS Conference, North Holland, Amsterdam, pp 589–603, 1981.

[53] K. Jörnsten, and M. Näsberg. A new Lagrangian relaxation approach to the generalized assignment problem. European Journal of Operational Research, 27(3), 313-323, 1986.

[54] A. Ceselli, G. Righini. A branch-and-price algorithm for the multilevel generalized assignment problem. Operations research, 54(6), 1172-1184, 2006.

[55] D.G. Cattrysse, M. Salomon, and L.N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. European Journal of Operational Research, 72(1), 167-174, 1994.

[56] L. A. N. Lorena, and M. G. Narciso. Relaxation heuristics for a generalized assignment problem. European Journal of Operational Research, 91(3), 600-610, 1996.

[57] S. Haddadi. Lagrangian decomposition based heuristic for the generalized assignment problem. Inf Syst Oper Res 37:392–402, 1999.

[58] M. G. Narciso, and L. A. N. Lorena. Lagrangean/surrogate relaxation for generalized assignment problems. European Journal of Operational Research, 114(1), 165-177,1999.

[59] S. Haddadi, and H. Ouzia. Effective algorithm and heuristic for the generalized assignment problem. European Journal of Operational Research, 153(1), 184-190, 2004.

[60] B. Lei, H. Zhu, and Y. Gningue, "Using Group Role Assignment to Solve Dynamic Vehicle Routing Problem," The 16th IEEE Int'l Conference on Networking, Sensing, and Control, Banff, Canada, May 9-11, pp. 104-109, 2019.

[61] M. M. Amini, and M. Racer. A rigorous computational comparison of alternative solution methods for the generalized assignment problem. Management Science, 40(7), 868-890, 1994.

[62] M. M. Amini, and M. Racer. A hybrid heuristic for the generalized assignment problem. European Journal of Operational Research, 87(2), 343-348, 1995.

[63] M. Yagiura, T. Yamaguchi, and T. Ibaraki. A variable depth search algorithm for the generalized assignment problem. In Meta-heuristics (pp. 459-471). Springer US, 1999.

[64] M. Yagiura, T. Yamaguchi, and T. Ibaraki. A variable depth search algorithm with branching search for the generalized assignment problem. Optimization Methods and Software, 10(2), 419-441, 1998.

[65] B. M. Lin, Y. S. Huang., and H. K. Yu. On the variable depth search heuristic for the linear-cost generalized assignment problem. International journal of computer mathematics, 77(4), 535-544, 2001.

[66] I. H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. Operations-Research-Spektrum, 17(4), 211-225, 1995.

[67] M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. INFORMS Journal on Computing, 16(2), 133-151, 2004.

[68] M. Yagiura, T. Ibaraki, and F. Glover. A path relinking approach with ejection chains for the generalized assignment problem. European journal of operational research, 169(2), 548-569, 2006.

[69] J. A. Dıaz, and E. Fernández. A tabu search heuristic for the generalized assignment problem. European Journal of Operational Research, 132(1), 22-38, 2001.

[70] P. C. Chu, & J. E. Beasley. A genetic algorithm for the generalized assignment problem. Computers & Operations Research, 24(1), 17-23, 1997.

[71] J.M. Wilson. A genetic algorithm for the generalized assignment problem. J Oper Res Soc 48:804–809, 1997.

[72] H. Feltl, G.R. Raidl. An improved hybrid genetic algorithm for the generalized assignment problem. In: SAC ’04; Proceedings of the 2004 ACM symposium on Applied computing. ACM Press, New York, pp 990–995, 2004.

[73] H.R. Lourenço, D. Serra. Adaptive approach heuristics for the generalized assignment problem. Technical Report 288, Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, 1998.

[74] L.A. Lorena, M.G. Narciso, and J. E. Beasley. A constructive genetic algorithm for the generalized assignment problem. Evolutionary Optimization, 5, 1-19, 2002.

[75] H. Zhu. Role-Based Collaboration and E-CARGO: Revisiting the Developments of the Last Decade Role-based collaboration is an emerging computational methodology that uses roles as the prim. IEEE Systems, Man, and Cybernetics Magazine. 1. 27-36, 2015.

[76] H. Zhu, and R. Alkins. Group Role Assignment. In 2009 International Symposium on Collaborative Technologies and Systems, 431–39. Baltimore, MD, USA: IEEE, 2009.

[77] Y. Xie. An o($n^{2.5}$) Algorithm: For Maximum Matchings in General Graphs. Journal of Applied Mathematics and Physics 06, no. 09: 1773–82, 2018.

[78] https://en.wikipedia.org/wiki/Assignment_problem

[79] H. W. Kuhn. The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly 2, no. 1–2: 83–97, March 1955.

[80] J. Munkres. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 1, 32-38, March 1957

[81] I. H. Toroslu, and G. Üçoluk. Incremental Assignment Problem. Information Sciences 177, no. 6 (March 15, 2007): 1523–29.

[82] A. Volgenant. An Addendum on the Incremental Assignment Problem. Information Sciences 178, no. 23 (December 1, 2008): 4583–4583.

[83] I. H. Toroslu, and G. Üçoluk. Authors' Response to "an Addendum on the Incremental Assignment Problem" by Volgenant. Vol. 178, 2008.

[84] R. K. Ahuja, T. L. Magnanti, J. B. Orlin. Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, New Jersey, 1993.

[85] R. E. Burkard, E. CELA. Handbook of Combinatorial Optimization, Supplement Volume A. Kluwer Academic Publishers, ch. Linear assignment problems and extensions, pp. 75–149, 1999.

[86] C. H. Papadimitriou, and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Dover Publications, 1998.

Appendix I

This appendix is the maximum running time and the minimum running time of Incremental Assignment Algorithm and Improved Incremental Assignment Algorithm in Case 1.

Time (ms) / Dimension	IIAA Max	IIAA Min	IAA Max	IAA Min
5	2.775900	0.809700	11.843000	3.605700
10	3.155300	1.007800	29.262300	5.532300
15	2.416800	0.783300	10.374100	6.320200
20	3.447700	0.740400	17.760300	9.486000
25	1.895801	0.830701	18.330300	10.197900
30	3.984400	1.172400	126.331600	17.426300
35	3.927100	1.014100	23.748100	16.285900
40	3.839500	1.153100	64.245200	28.302800
45	3.418000	1.304300	49.567500	28.510500
50	2.508400	1.216200	53.943300	39.909500
55	2.535800	1.474100	59.939000	47.028700
60	3.967800	1.262800	117.148000	50.581900
65	4.301000	1.554300	100.045200	18.575000
70	4.982300	1.660900	115.891799	75.239301
75	4.569700	1.582000	130.493100	63.174000
80	5.879900	3.174800	105.043500	61.280800
85	3.989800	2.301400	100.358800	69.800300
90	3.914600	2.107500	121.074600	84.413500
95	4.487800	2.397500	117.880600	91.999500
100	6.524400	2.614800	117.311600	92.163900

Appendix II

This appendix is the maximum running time and the minimum running time of Incremental Assignment Algorithm and Improved Incremental Assignment Algorithm in Case 2.

Time (ms) / Dimension	IIAA Max	IIAA Min	IAA Max	IAA Min
5	2.486000	0.754800	7.339600	3.118100
10	3.084600	0.961000	25.906100	6.381400
15	2.346800	0.965800	14.566900	7.281400
20	2.679800	0.712900	22.630200	8.092000
25	1.765600	0.838500	38.868401	13.083299
30	3.493000	0.936900	47.798400	19.708000
35	2.515800	0.912000	28.608900	19.834100
40	3.031899	1.168000	79.442600	34.455000
45	2.310200	1.167000	46.856000	31.639800
50	3.667600	1.391700	131.671800	36.891700
55	2.380500	1.421900	62.886900	43.799300
60	3.196000	1.261800	108.415700	58.537300
65	4.993500	1.402700	146.824500	65.585900
70	4.617400	1.465400	110.288201	94.628900
75	7.532800	3.194000	117.590000	67.875500
80	4.934900	3.021400	106.298400	59.418300
85	4.305500	2.442900	133.620500	65.973100
90	4.233100	2.301500	119.383700	83.284900
95	4.585900	1.983400	114.241300	94.118200
100	5.977300	3.177000	130.864300	97.024000

Appendix III

This appendix is the maximum running time and the minimum running time of Incremental Assignment Algorithm and Improved Incremental Assignment Algorithm in Case 3.

Time (ms) / Dimension	IIAA Max	IIAA Min	IAA Max	IAA Min
5	10.051900	1.170300	9.688700	4.344700
10	21.439200	1.068400	20.592600	9.294100
15	12.250700	1.141100	12.566400	8.135600
20	24.431200	1.483900	30.004400	10. 442700
25	25.724999	0.98,500	27.136800	10.082001
30	39.386900	1.218300	44.763300	21.523700
35	25.798600	2.613300	28,090800	17.720900
40	72.103400	27.782800	67.590700	27.915900
45	38.080800	25.299400	39.573800	28.396700
50	49.055900	3.807700	66.579600	42.349600
55	68.793900	2.380500	63.291800	44.966700
60	116.272900	2.486500	138.822900	57.100400
65	110.295600	61.285200	103.969100	67.856300
70	107.151899	3.860099	172.658000	86.639501
75	88.357300	62.410200	108.533400	67.468700
80	91.485100	4.061300	126.127100	81.382000
85	103.893300	81.419300	125.805100	91.329700
90	161.647200	4.233100	127.307100	87.790700
95	107.744600	2.397500	127.115600	92.160100
100	122.337800	4.048500	123.669300	93.273000